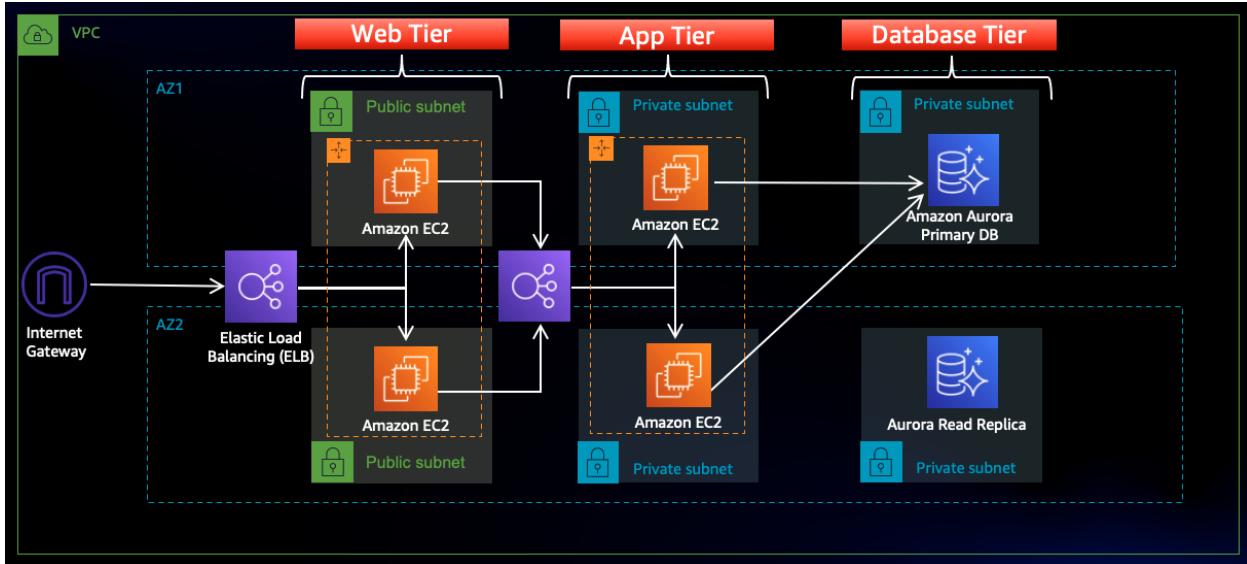


# Architecture Diagram for AWS



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.

This screenshot shows a web browser window with the following details:

- Title Bar:** Project2 - Deploy a 3 Tier A, AWS Three Tier Web Application
- Address Bar:** catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US
- Header:** aws workshop studio
- Main Content:**
  - AWS Three Tier Web Architecture**
  - Description:** This workshop is a hands-on walk through of a three-tier web architecture in AWS. We will be manually creating the necessary network, security, app, and database components and configurations in order to run this architecture in an available and scalable manner.
  - Audience:** Although this is an introductory level workshop, it is intended for those who have a technical role. The assumption is that you have at least some foundational AWS knowledge around VPC, EC2, RDS, S3, ELB and the AWS Console.
  - Workshop Duration:** The expected duration of this workshop is around 3 hours.
  - Clean-Up:**
- Footer:** © 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy policy Terms of use Cookie preferences

This screenshot shows a web browser window with the following details:

- Title Bar:** Project2 - Deploy a 3 Tier Archi..., AWS Three Tier Web Application, Siddhartha082/AWS-3-Tier-w... | +
- Address Bar:** catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part0/code
- Header:** aws workshop studio
- Main Content:**
  - AWS Three Tier Web Architecture**
  - Download Code from Github**
  - Download the code from [this repository](https://github.com/aws-samples/aws-three-tier-web-architecture-workshop.git) 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
```
  - Navigation:** Previous, Next
- Footer:** © 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy policy Terms of use Cookie preferences

Project2 - Deploy a 3 Tier Arch | AWS Three Tier Web Application | Siddhartha082/AWS-3-Tier-w | +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part0/s3bucketcreation

aws workshop studio

## 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 service page. On the left, there's a sidebar with 'Buckets' selected. The main area has a heading 'Account snapshot' with metrics: Total storage (112.1 GB), Object count (3.8 M), and Avg. object size (30.6 KB). Below this is a table titled 'Buckets (47) Info' showing a list of existing buckets. At the bottom right of the table, there is a prominent orange 'Create bucket' button, which is highlighted with a red box.

The screenshot shows the AWS S3 service page. On the left, there's a sidebar with 'Buckets' selected. The main area has a heading 'Account snapshot' with metrics: Total storage (112.1 GB), Object count (3.8 M), and Avg. object size (30.6 KB). Below this is a table titled 'Buckets (47) Info' showing a list of existing buckets. At the bottom right of the table, there is a prominent orange 'Create bucket' button, which is highlighted with a red box.

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.

**Create bucket** Info

Buckets are containers for data stored in S3. Learn more [\[?\]](#)

### General configuration

Bucket name  ←

AWS Region  ←

Copy settings from existing bucket - optional  
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

### Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)  
All objects in this bucket are owned by this account.  
Access to this bucket and its objects is specified using only policies.

ACLs enabled  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership  
Bucket owner enforced

## IAM EC2 Instance Role Creation

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

**Identity and Access Management (IAM)**

Search IAM

Dashboard

Access management

User groups

Users

**Roles** ←

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analyzers

Settings

Credential report

Organization activity

Service control policies (SCPs)

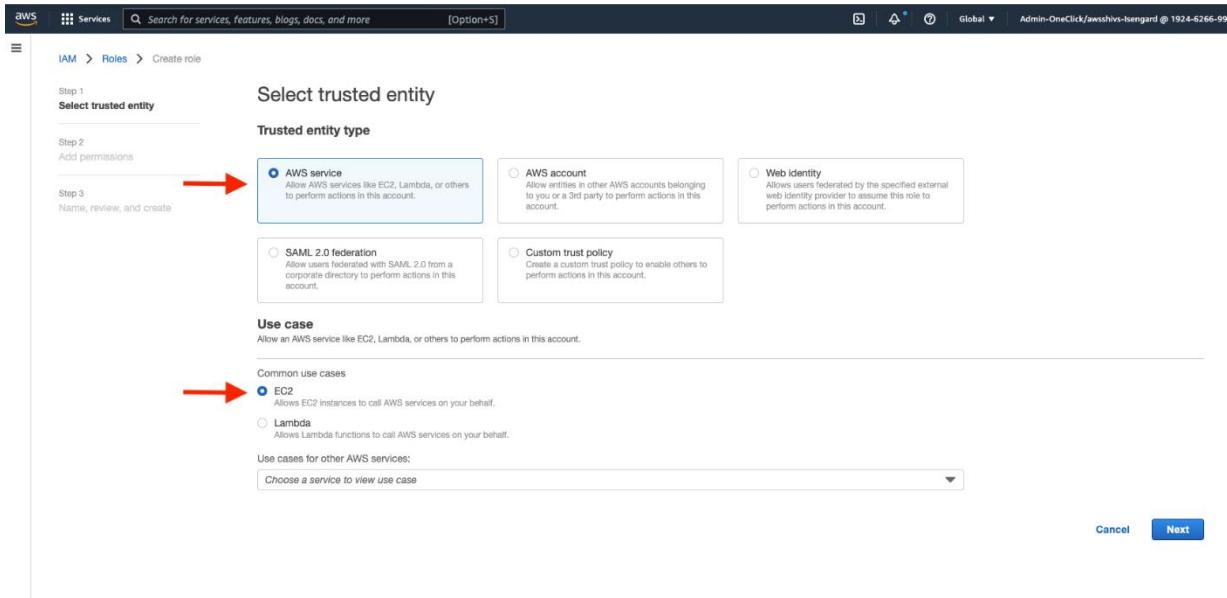
**IAM > Roles**

**Roles (190)** Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

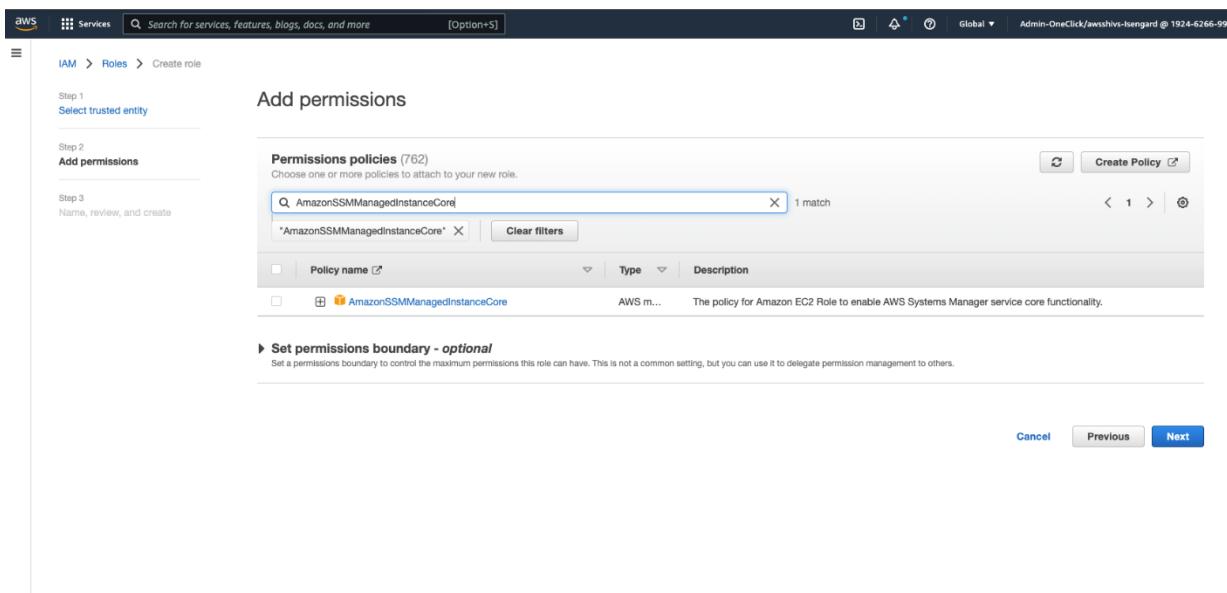
<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	AccessAnalyzerTrustedService	Account: 121925031476	-
<input type="checkbox"/>	Admin-OneClick	Account: 727820809195	28 minutes ago
<input type="checkbox"/>	Amazon-GlueServiceRoleForSSM	AWS Service: glue	7 hours ago
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-202101002T123844	AWS Service: sagemaker	37 minutes ago
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-20210322T12907	AWS Service: sagemaker	36 minutes ago
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-20210603T120053	AWS Service: sagemaker	1 hour ago
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-20210608T171120	AWS Service: sagemaker	272 days ago
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-20210628T150369	AWS Service: sagemaker	37 minutes ago
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-20210902T110148	AWS Service: sagemaker	-
<input type="checkbox"/>	AmazonSageMaker-ExecutionRole-20210914T135287	AWS Service: sagemaker	45 minutes ago
<input type="checkbox"/>	AmazonSageMakerServiceCatalogProductsLaunchRole	AWS Service: servicecatalog	168 days ago
<input type="checkbox"/>	AmazonSageMakerServiceCatalogProductsUseRole	AWS Service: states, and 9 more. <a href="#">[?]</a>	29 minutes ago
<input type="checkbox"/>	AmazonSSMRoleForAutomationAssumeQuickSetup	AWS Service: ssm	-
<input type="checkbox"/>	AmazonSSMRoleForInstancesQuickSetup	AWS Service: ec2	-

Select EC2 as the trusted entity.



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**



1. Give your role a name, and then click **Create Role**.

The screenshot shows a browser window titled 'catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part1'. The page is part of the 'aws workshop studio' and displays the 'AWS Three Tier Web Architecture' under 'Part 1: Networking and Security'. A note states: '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.' Below this, a 'Learning Objectives:' section lists: 'Create an isolated network with the following components: VPC, Subnets, Route Tables, Internet Gateway, NAT gateway, Security Groups'. At the bottom of the page, there is a footer with links to 'Privacy policy', 'Terms of use', and 'Cookie preferences'. The browser's taskbar at the bottom shows various open tabs and system icons.

## VPC and Subnets

### 1. VPC Creation

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

**Your VPCs (1) Info**

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options
vpc-01c95becccc57d54	Available	172.31.0.0/16	-	-	dopt-0d05k

Select a VPC above

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

**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.

**VPC settings**

Resources to create: **VPC only** (selected)

Name tag - optional: awsthetierworkshop

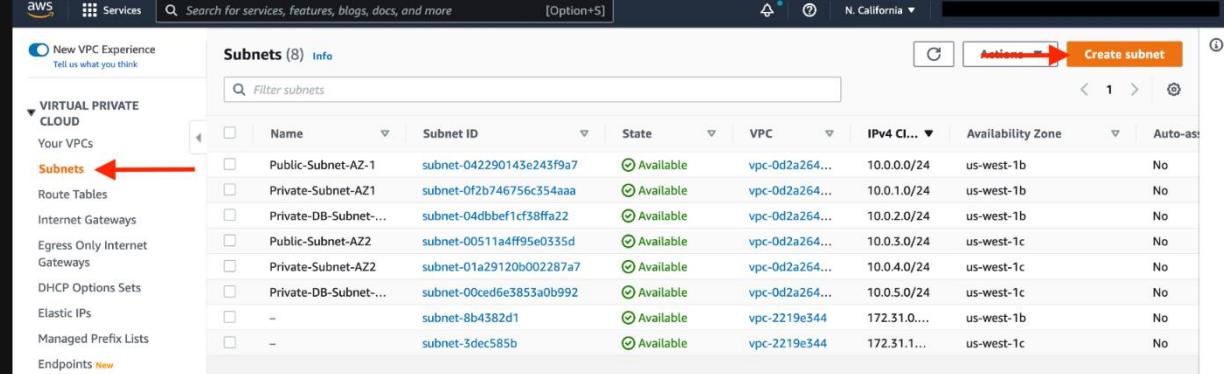
IPv4 CIDR: 10.0.0.0/16

Tags: Key: Name, Value: awsthetierworkshop

Create VPC

## 2. Subnet Creation

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

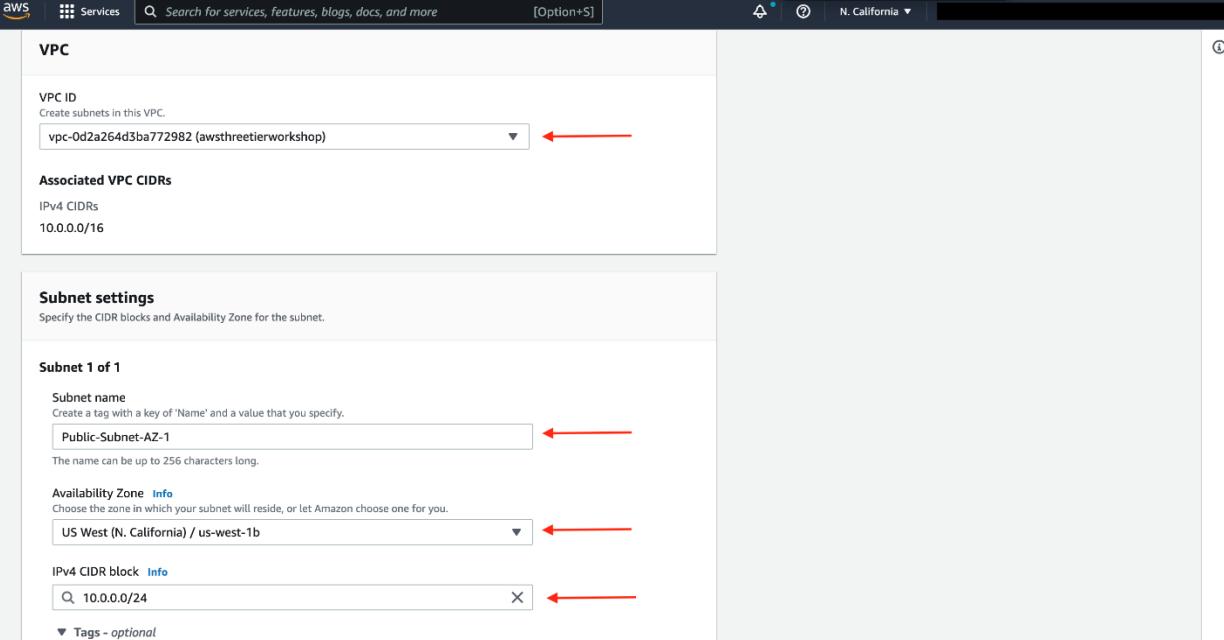


The screenshot shows the AWS VPC Subnets page. On the left, there's a sidebar with 'New VPC Experience' and a list of VPC-related services: VIRTUAL PRIVATE CLOUD, Your VPCs, Subnets (highlighted with a red arrow), Route Tables, Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, Elastic IPs, Managed Prefix Lists, and Endpoints. The main area displays a table of subnets with columns: Name, Subnet ID, State, VPC, IPv4 CIDR Range, Availability Zone, and Auto-assign. There are 8 subnets listed, all in an available state across different VPCs and availability zones.

2. We will need **six subnets** across **two availability zones**. That means that **three subnets** will be in one availability zone, and three subnets will be in another zone. Each subnet in one availability zone will correspond to one layer of our three tier architecture. Create each of the 6 subnets by specifying the VPC we created in part 1 and then choose a name, availability zone, and appropriate CIDR range for each of the subnets.

*NOTE: It may be helpful to have a naming convention that will help you remember what each subnet is for. For example in one AZ you might have the following: Public-Web-Subnet-AZ-1, Private-App-Subnet-AZ-1, Private-DB-Subnet-AZ-1.*

*NOTE: Remember, your CIDR range for the subnets will be subsets of your VPC CIDR range.*



The screenshot shows the 'Create subnet' wizard. Step 1: VPC settings. It shows the VPC ID dropdown set to 'vpc-0d2a264d3ba772982 (awsthetierworkshop)'. Under 'Associated VPC CIDRs', it shows 'IPv4 CIDRs' and '10.0.0.0/16'. In the 'Subnet settings' section, it shows 'Subnet 1 of 1' with 'Subnet name' set to 'Public-Subnet-AZ-1', 'Availability Zone' set to 'US West (N. California) / us-west-1b', and 'IPv4 CIDR block' set to '10.0.0.0/24'.

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

Name	Subnet ID	State	VPC	IPv4 CIDR	Availability Zone
Public-Subnet-AZ-1	subnet-042290143e243f9a7	Available	vpc-0d2a264...	10.0.0.0/24	us-west-1b
Private-Subnet-AZ1	subnet-0f2b746756c354aaa	Available	vpc-0d2a264...	10.0.1.0/24	us-west-1b
Private-DB-Subnet-AZ1	subnet-04dbbef1cf38fa22	Available	vpc-0d2a264...	10.0.2.0/24	us-west-1b
Public-Subnet-AZ2	subnet-00511a4ff95e0335d	Available	vpc-0d2a264...	10.0.3.0/24	us-west-1c
Private-Subnet-AZ2	subnet-01a29120b002287a7	Available	vpc-0d2a264...	10.0.4.0/24	us-west-1c
Private-DB-Subnet-AZ2	subnet-00ced6e3853a0b992	Available	vpc-0d2a264...	10.0.5.0/24	us-west-1c

## Internet Connectivity

**1. 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**.

Internet gateways (1) Info

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-6cfb000b	Attached	vpc-2219e344	192462669998

Select an internet gateway above

Project2 - Deploy a 3 Tier Arch | Editing AWS-3-Tier-web-Arch | AWS Three Tier Web Application

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part1/internetconnectivity

aws workshop studio

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

VPC > Internet gateways > Create internet gateway

### Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

**Internet gateway settings**

Name tag  
Creates a tag with a key of 'Name' and a value that you specify.  
 ←

**Tags - optional**  
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.

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="three-tier-igw"/> <span style="color:red">←</span>
<span style="border:1px solid black; padding:2px;">Remove</span>	

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Windows Type here to search 🍒 27°C Mostly cloudy ENG 16:46 09-07-2024

Project2 - Deploy a 3 Tier Arch | Editing AWS-3-Tier-web-Arch | AWS Three Tier Web Application

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part1/internetconnectivity

aws workshop studio

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.

VPC > Internet gateways > igw-04dfd082c3c213815

### igw-04dfd082c3c213815 / three-tier-igw

**Details Info**

Internet gateway ID <input type="text" value="igw-04dfd082c3c213815"/>	State <span style="color:gray;">Detached</span>	VPC ID -	Owner 192462669998
---	--	-------------	-----------------------

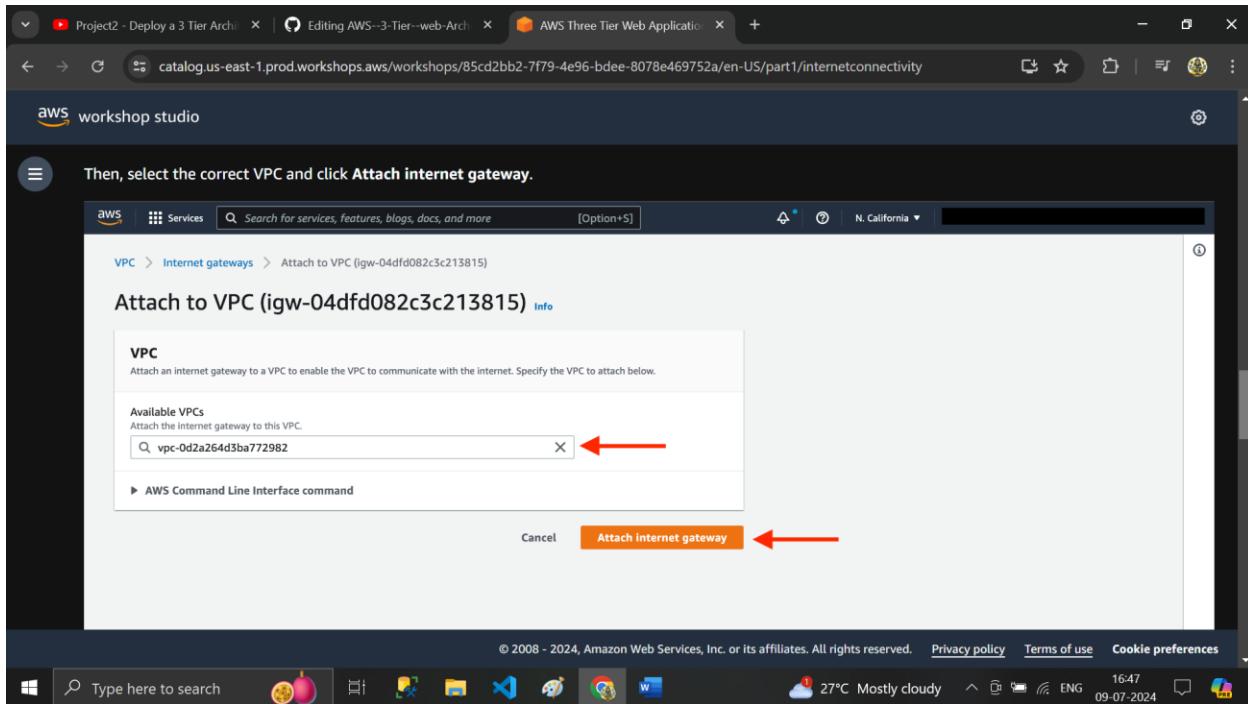
**Tags**  
 ←

**Actions ▾**

- Attach to VPC →
- Detach from VPC
- Manage tags
- Delete

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Windows Type here to search 🍒 27°C Mostly cloudy ENG 16:47 09-07-2024



## 2. NAT Gateway

1. 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**.

**New VPC Experience** Tell us what you think

**NAT gateways Info**

**Create NAT gateway** ↑

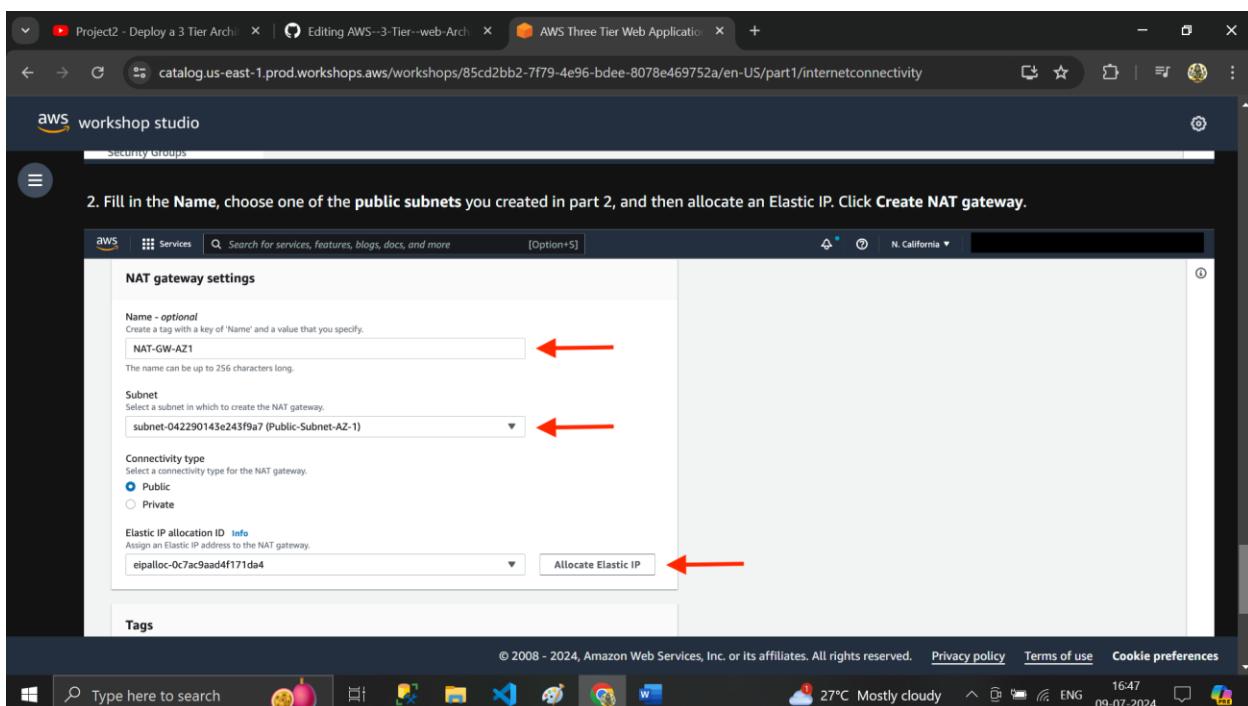
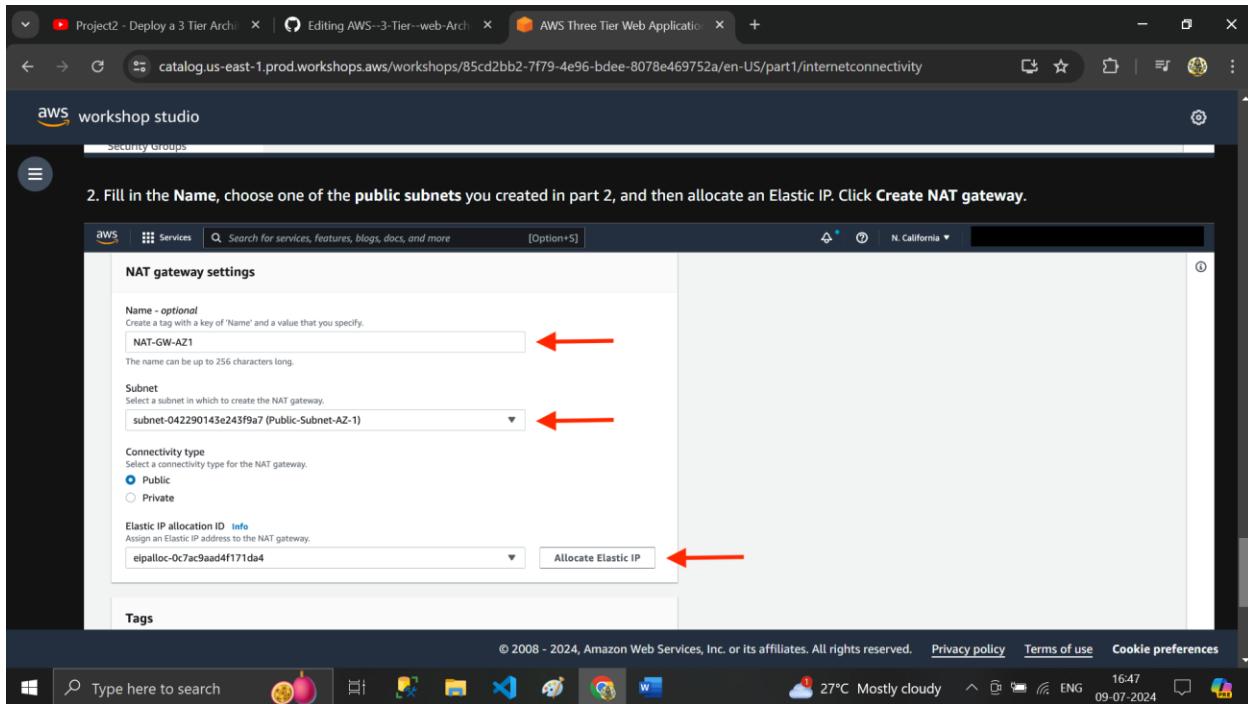
Name	NAT gateway ID	Connectivit...	State	State message	Elastic IP address
Select a NAT gateway					

**VIRTUAL PRIVATE CLOUD**

- Your VPCs
- Subnets
- Route Tables
- Internet Gateways
- Egress Only Internet Gateways
- DHCP Options Sets
- Elastic IPs
- Managed Prefix Lists
- Endpoints New
- Endpoint Services
- NAT Gateways** ←
- Peering Connections

**SECURITY**

- Network ACLs
- Security Groups



## Routing Configuration

1. 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.

Screenshot of the AWS VPC Route Tables page:

The left sidebar shows the VPC Dashboard with the "Route Tables" section highlighted by a red arrow.

The main content area displays a table titled "Route tables (2) Info". The table has columns: Name, Route table ID, Explicit subnet associat..., Edge associations, Main, and VPC. Two route tables are listed:

Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC
-	rtb-04aef0875986d8ca0	-	-	Yes	vpc-0d2a264d3ba772982   aw...
-	rtb-fa5f2e9c	-	-	Yes	vpc-2219e344

A red arrow points to the "Create route table" button in the top right corner of the table header.

Screenshot of the "Create route table" wizard:

The title bar says "Create route table Info". A note below states: "A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection."

**Route table settings**

**Name - optional**: PublicRouteTable (highlighted by a red arrow)

**VPC**: vpc-0d2a264d3ba772982 (awsthiereiworkshop) (highlighted by a red arrow)

**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.

Key: Name, Value: PublicRouteTable, Remove, Add new tag (highlighted by a red arrow)

You can add 49 more tags.

Cancel, Create route table (highlighted by a red arrow)

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. California ⓘ

### Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

#### 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.

Key	Value - <i>optional</i>
<input type="text" value="Name"/>	<input type="text" value="PublicRouteTable"/>
<a href="#">Remove</a>	
<a href="#">Add new tag</a>	

You can add 49 more tags.

[Cancel](#) **Create route table** ←

Project2 - Deploy a 3 Tier Arch! | Editing AWS-3-Tier-web-Arch | AWS Three Tier Web Application +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part1/routing

aws workshop studio

**Details** Info

Route table ID	Main	Explicit subnet associations	Edge associations
<a href="#">rtb-0a603176bd4a1da5</a>	<input checked="" type="checkbox"/> No	-	-
VPC	Owner ID	192462669998	
vpc-0d2a264d3ba772982   awsthetierworkshop			

**Routes** Edit routes

Routes (1)				
<input type="text" value="Filter routes"/> Both < 1 > ⌂				
Destination	Target	Status	Propagated	
10.0.0.0/16	local	<input checked="" type="checkbox"/> Active	No	

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Type here to search → Very high UV 16:50 09-07-2024 ENG

3. 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.

4. 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**.

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0a6063176bd4a1da5	No	-	-
VPC	Owner ID		
vpc-0d2a264d3ba772982   awsthreetierworkshop	192462669998		

**Explicit subnet associations (0)**

**Edit subnet associations**

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

VPC > Route tables > rtb-0a6063176bd4a1da5 > Edit subnet associations

### Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (2/6)						
	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID	
<input type="checkbox"/>	Private-DB-Subnet-AZ1	subnet-04dbbef1cf38ffa22	10.0.2.0/24	-	Main (rtb-04aef0875986d8ca0)	
<input checked="" type="checkbox"/>	Public-Subnet-AZ-1	subnet-042290143e243f9a7	10.0.0.0/24	-	Main (rtb-04aef0875986d8ca0)	
<input type="checkbox"/>	Private-Subnet-AZ1	subnet-0f2b746756c354aa	10.0.1.0/24	-	Main (rtb-04aef0875986d8ca0)	
<input type="checkbox"/>	Private-DB-Subnet-AZ2	subnet-00ced6e3853a0b992	10.0.5.0/24	-	Main (rtb-04aef0875986d8ca0)	
<input type="checkbox"/>	Private-Subnet-AZ2	subnet-01a29120b002287a7	10.0.4.0/24	-	Main (rtb-04aef0875986d8ca0)	
<input checked="" type="checkbox"/>	Public-Subnet-AZ2	subnet-00511a4ff95e0335d	10.0.3.0/24	-	Main (rtb-04aef0875986d8ca0)	

**Selected subnets**

subnet-00511a4ff95e0335d / Public-Subnet-AZ2 X | subnet-042290143e243f9a7 / Public-Subnet-AZ-1 X

**Save associations**

Project2 - Deploy a 3 Tier Archi... | Editing AWS 3-Tier Web Application | AWS Three Tier Web Application

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part1/routing

aws workshop studio

4. 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.

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

**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.  
 ←

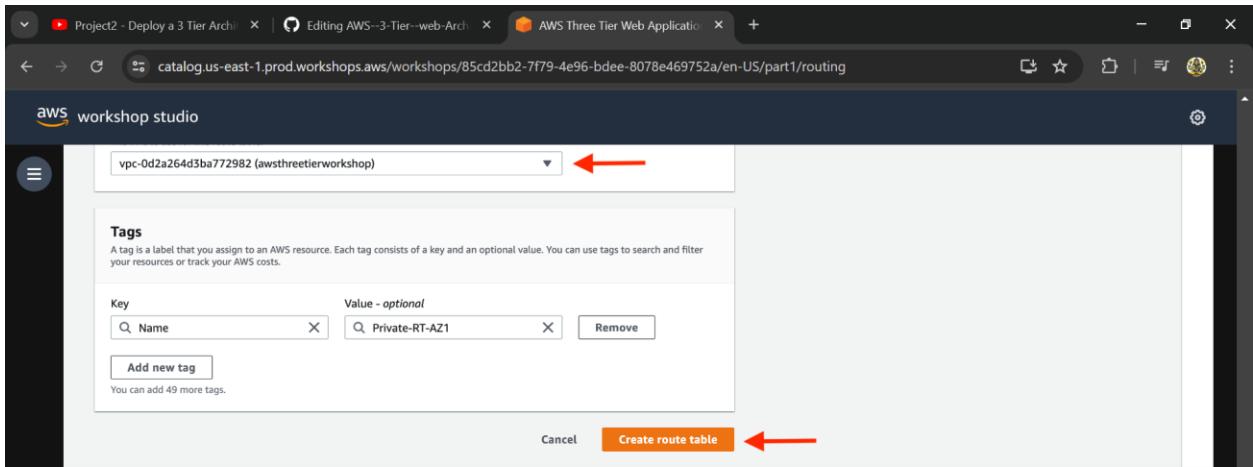
**Tags**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Type here to search ← Earnings upcoming 16:52 ENG 09-07-2024

The screenshot illustrates the process of creating a route table and adding a route to it.

**Step 1: Creating a Route Table**



AWS workshop studio

vpc-0d2a264d3ba772982 (awsthetierworkshop)

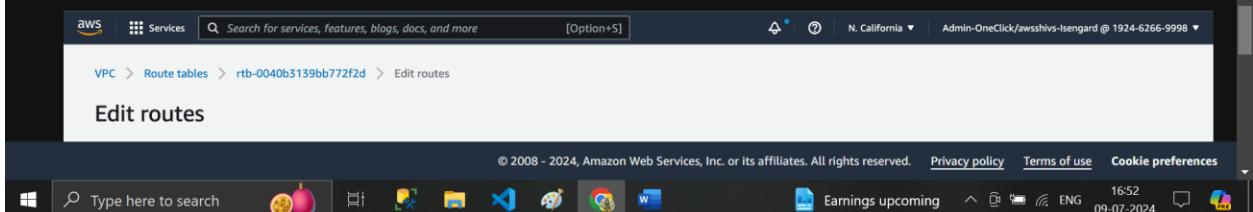
**Tags**

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.

Key: Name Value - optional: Private-RT-AZ1 Remove Add new tag You can add 49 more tags.

Create route table

**Step 2: Adding a Route to the Route Table**



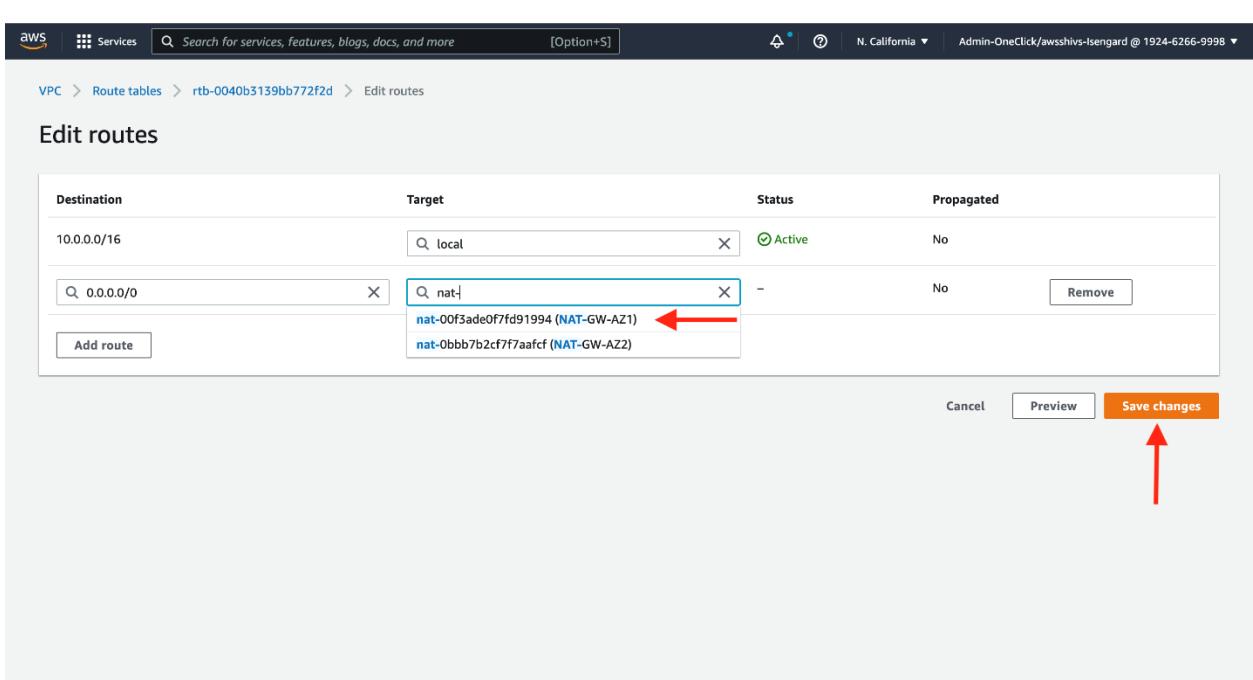
aws Services Search for services, features, blogs, docs, and more [Option+S] N. California Admin-OneClick/awsshivs-tsengard @ 1924-6266-9998

VPC > Route tables > rtb-0040b3139bb772f2d > Edit routes

Edit routes

Type here to search Earnings upcoming 16:52 09-07-2024

**Step 3: Final View of the Route Table Configuration**



aws Services Search for services, features, blogs, docs, and more [Option+S] N. California Admin-OneClick/awsshivs-tsengard @ 1924-6266-9998

VPC > Route tables > rtb-0040b3139bb772f2d > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	nat-00f3ade0f7fd91994 (NAT-GW-AZ1)	-	No

Add route

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.

**Edit subnet associations**

Change which subnets are associated with this route table.

**Available subnets (1/6)**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
Private-DB-Subnet-AZ1	subnet-04dbbef1cf38ffa22	10.0.2.0/24	-	Main (rtb-04aef0875986d8ca0)
Public-Subnet-AZ-1	subnet-042290143e243f9a7	10.0.0.0/24	-	rtb-0a6063176bd4a1da5 / PublicRouteTable
<input checked="" type="checkbox"/> Private-Subnet-AZ1	subnet-0f2b746756c354aaa	10.0.1.0/24	-	Main (rtb-04aef0875986d8ca0)
Private-DB-Subnet-AZ2	subnet-00ced6e3853a0b992	10.0.5.0/24	-	Main (rtb-04aef0875986d8ca0)
Private-Subnet-AZ2	subnet-01a29120b002287a7	10.0.4.0/24	-	Main (rtb-04aef0875986d8ca0)
Public-Subnet-AZ2	subnet-00511a4ff95e0335d	10.0.3.0/24	-	rtb-0a6063176bd4a1da5 / PublicRouteTable

**Selected subnets**

subnet-0f2b746756c354aaa / Private-Subnet-AZ1

**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 Management Console with the VPC service selected. In the left navigation pane, under the 'SECURITY' section, 'Security Groups' is highlighted with a red arrow. On the main page, the title 'Security Groups (1) Info' is displayed above a table. The table has columns: Name, Security group ID, Security group name, VPC ID, Description, and Owner. A single row is shown with the following values: Name is blank, Security group ID is sg-0bf79a59048818994, Security group name is default, VPC ID is vpc-0d2a264d3ba772982, Description is 'default VPC security gr...', and Owner is 192462669998. At the top right of the table, there is a 'Create security group' button with a red arrow pointing to it.

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**.

The screenshot shows the 'Create security group' wizard. The first step, 'Basic details', is completed with the following information:

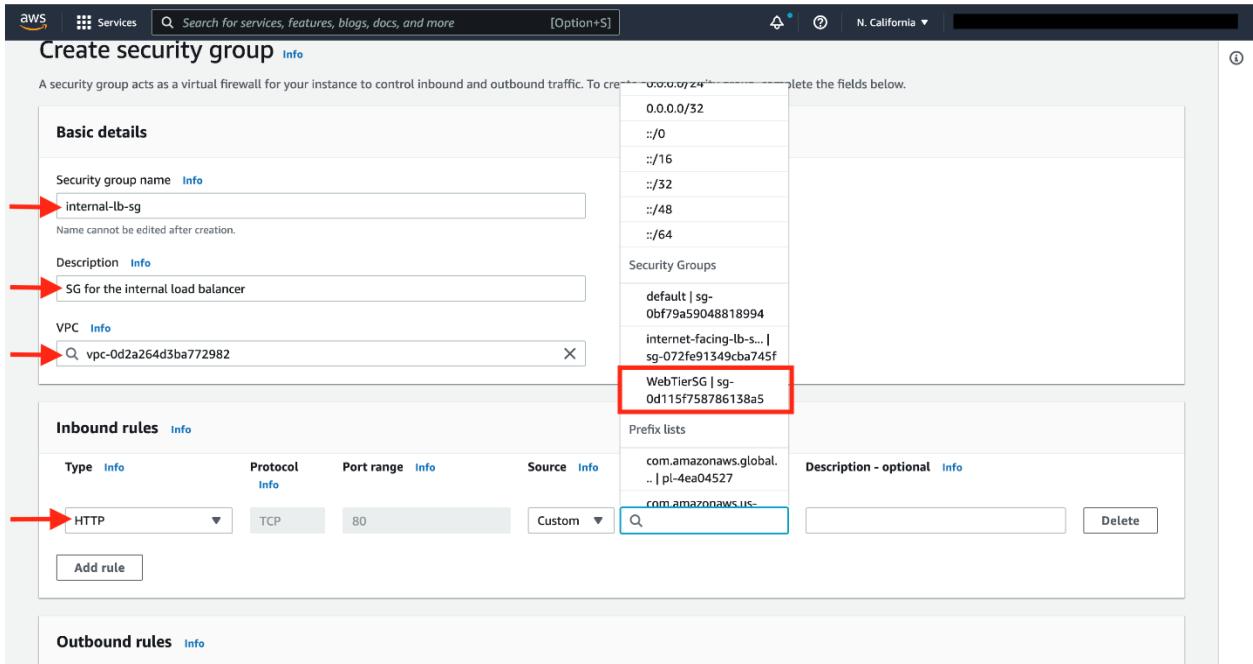
- Security group name: **internet-facing-lb-sg** (highlighted with a red arrow)
- Description: **External load balancer security group** (highlighted with a red arrow)
- VPC: **vpc-0d2a264d3ba772982** (highlighted with a red arrow)

Below this, the 'Inbound rules' section is shown, which is currently empty. An 'Add rule' button is visible at the bottom left of this section.

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.

The screenshot shows the AWS Services dashboard with the search bar set to "Search for services, features, blogs, docs, and more". The "N. California" region is selected. A modal window titled "Create New Security Group" is open. In the "Basic details" section, the "Security group name" field contains "WebTierSG" (with a red arrow pointing to it), and the "Description" field contains "SG for the Web Tier" (with a red arrow pointing to it). The "VPC" dropdown shows "vpc-0d2a264d3ba772982" (with a red arrow pointing to it). In the "Inbound rules" section, there are two entries. Both entries have "Type" set to "HTTP" (indicated by red arrows) and "Protocol" set to "TCP". The first entry has "Port range" set to "80" and "Source" set to "My IP". The second entry has "Port range" set to "80" and "Source" set to "Custom", with a dropdown menu open showing "sg-072fe91349cba745f" (highlighted with a blue box).

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.



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.

The screenshot shows the 'Create security group' wizard in the AWS Management Console. In the 'Basic details' section, the security group name is set to 'PrivateInstanceSG', its description is 'SG for our private app tier sg', and it is associated with the VPC 'vpc-0d2a264d3ba772982'. In the 'Inbound rules' section, two custom TCP rules are defined, both allowing port 4000 from 'My IP'. A red arrow points to the 'Source' dropdown for the second rule. On the right side of the screen, a sidebar lists existing security groups: 'default | sg-0bf79a59048818994', 'internet-facing-lb-s... | sg-072fe91349cba745f', 'WebTierSG | sg-0d115f758786138a5', and 'internal-lb-sg | sg-022b4036ffff83a40'. The last item is highlighted with a red box. A red arrow also points to the 'Description - optional' field.

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).

The screenshot shows the 'Create security group' wizard in the AWS Management Console. In the 'Basic details' section, the security group name is 'DBSG', its description is 'SG for our databases', and it is associated with the VPC 'vpc-0d2a264d3ba772982'. In the 'Inbound rules' section, a single rule is defined: a MySQL/Aurora rule on port 3306 from the 'PrivateInstanceSG' security group. A red arrow points to the 'Source' dropdown for this rule. The 'Outbound rules' section is partially visible at the bottom.

This screenshot shows a browser window with three tabs open: 'Project2 - Deploy a 3 Tier Archi...', 'Editing AWS--3-Tier--web-Arch...', and 'AWS Three Tier Web Application'. The main content area is titled 'aws workshop studio' and 'Part 2: Database Deployment'. It includes a section about deploying the database layer of a three-tier architecture and a 'Learning Objectives' section with bullet points: 'Deploy Database Layer', 'Subnet Groups', and 'Multi-AZ Database'. At the bottom, there's a footer with copyright information and links to 'Privacy policy', 'Terms of use', and 'Cookie preferences'. The browser's address bar shows the URL 'catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part2'. The operating system taskbar at the bottom shows various pinned icons and the date '09-07-2024'.

## Subnet Groups

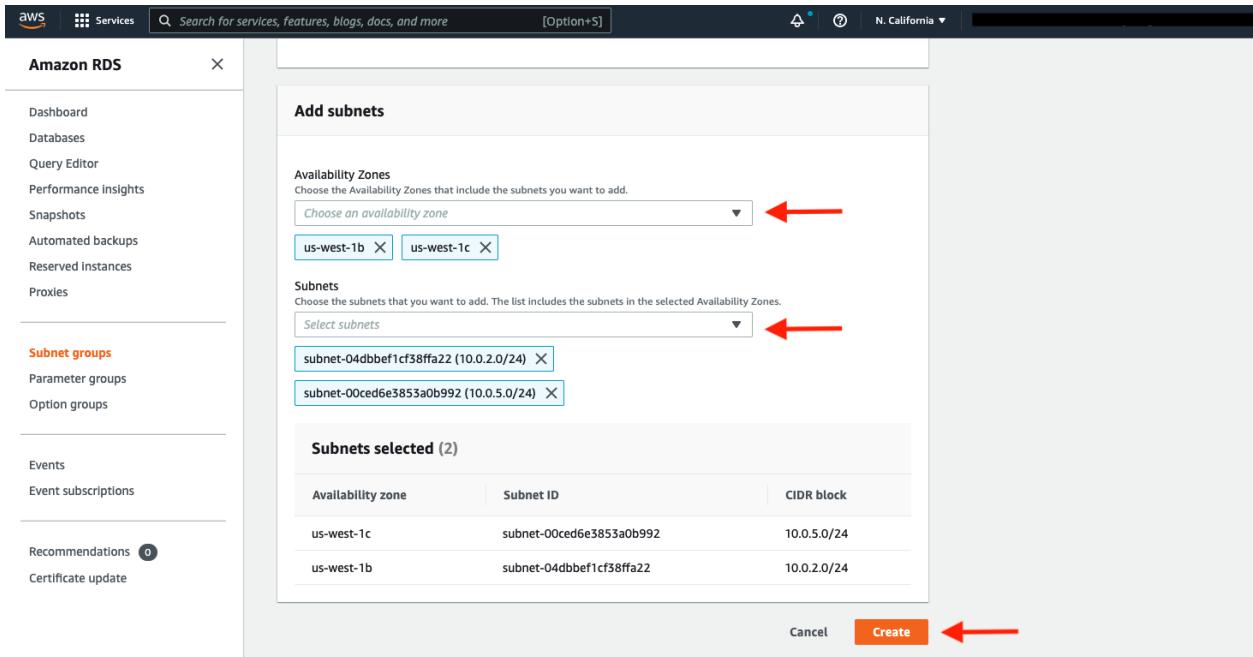
1. Navigate to the RDS dashboard in the AWS console and click on **Subnet groups** on the left hand side. Click **Create DB subnet grou**

This screenshot shows the 'Amazon RDS' service dashboard. The left sidebar has a 'Subnet groups' link highlighted with a red arrow. The main content area is titled 'Subnet groups (0)' and features a 'Create DB subnet group' button. A red arrow points to this button. Below it, a message says 'No db subnet groups' and 'You don't have any db subnet groups.' There is also a 'Create DB subnet group' button at the bottom of the list table.

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

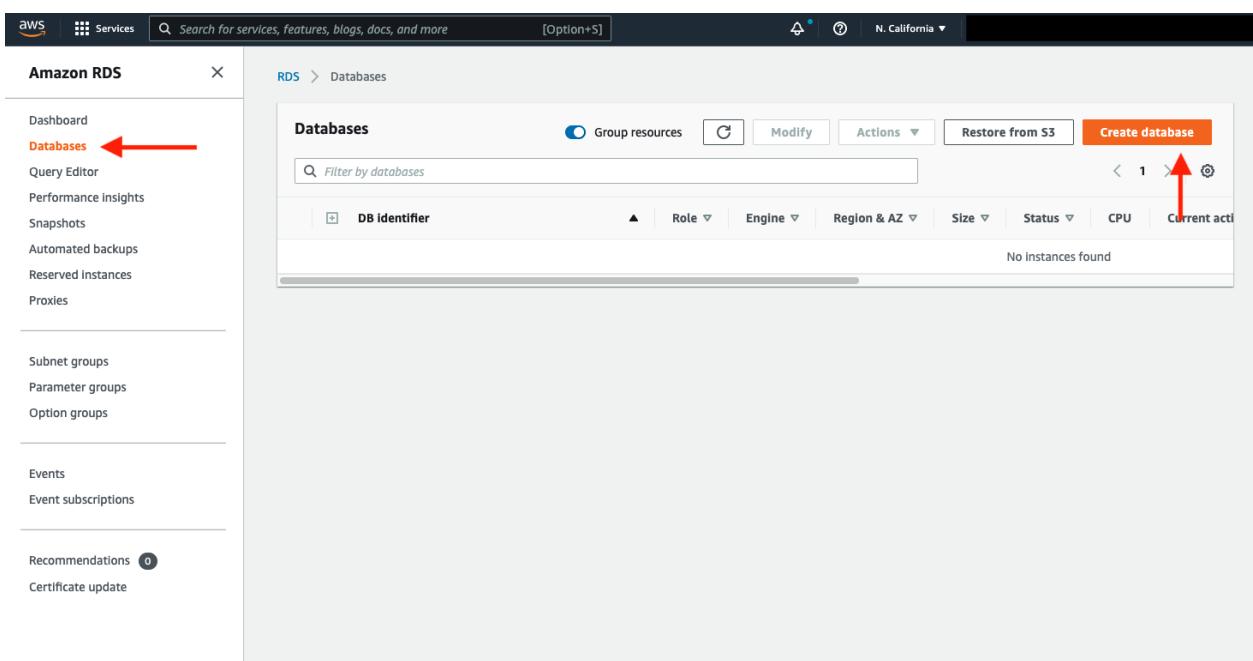
The screenshot shows the 'Create DB subnet group' page in the Amazon RDS console. On the left, there's a sidebar with links like Dashboard, Databases, Query Editor, etc., and a 'Subnet groups' section which is currently selected. The main area has a title 'Create DB subnet group' and a sub-section 'Subnet group details'. It includes fields for 'Name' (containing 'three-tier-db-subnet-group'), 'Description' (containing 'Subnet group for the database layer of the architecture.'), and 'VPC' (set to 'awsthreetierworkshop (vpc-0d2a264d3ba772982)'). Below this is another section titled 'Add subnets' with a 'Availability Zones' dropdown labeled 'Choose an availability zone'.

3. 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.

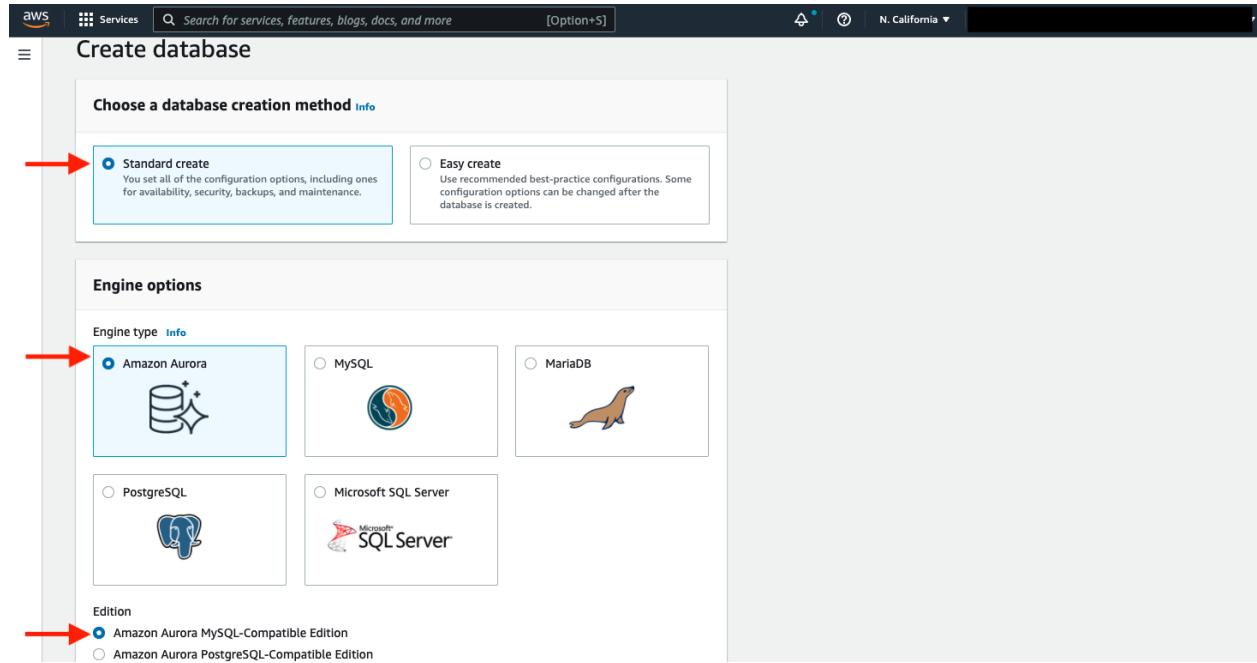


# Database Deployment

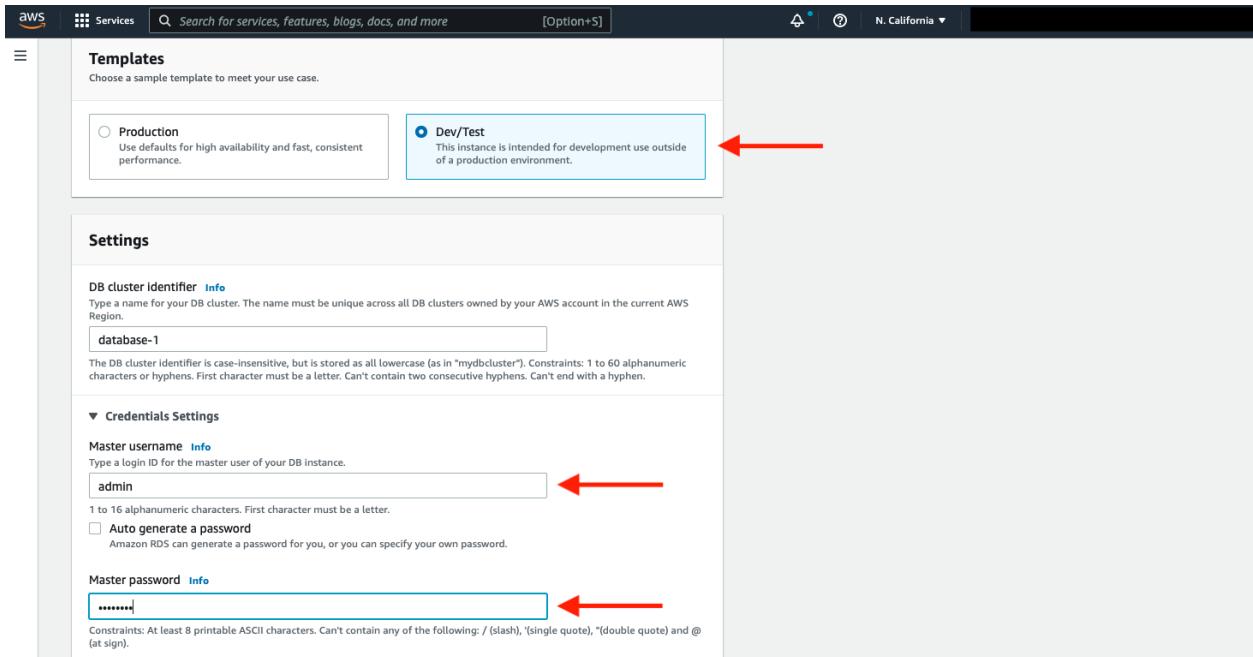
1. Navigate to **Databases** on the left hand side of the RDS dashboard and click **Create database**.



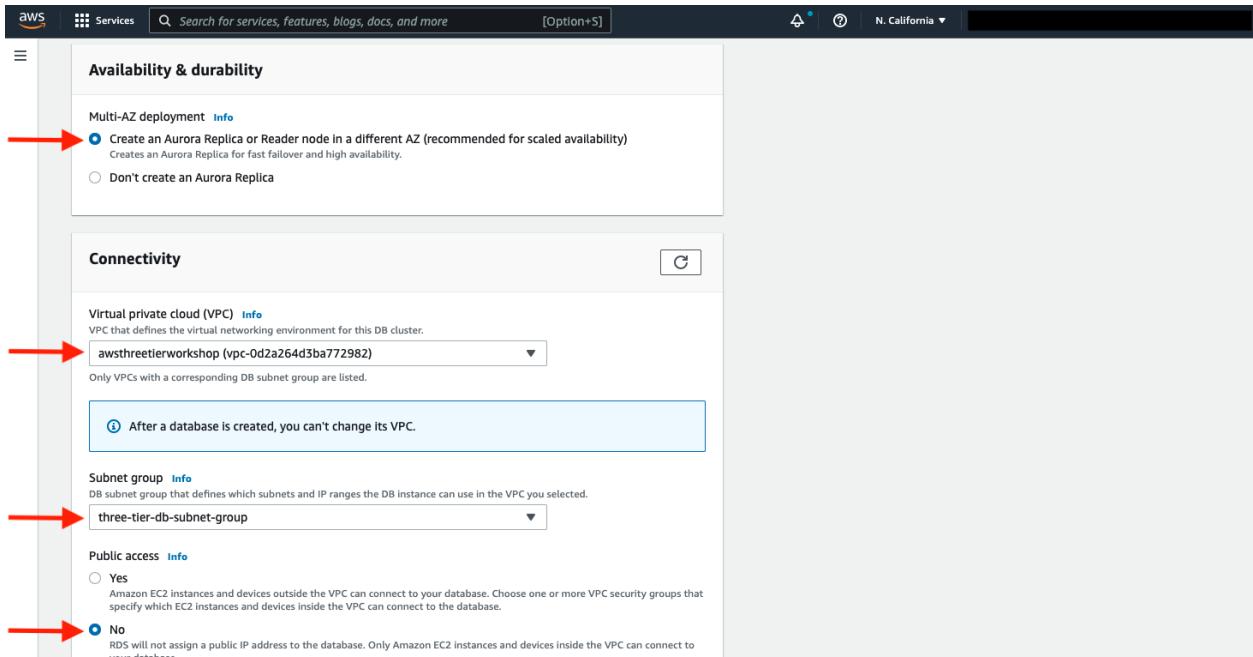
2. We'll now go through several configuration steps. Start with a **Standard create** for this **MySQL-Compatible Amazon Aurora** database. Leave the rest of the defaults in the **Engine options** as default.



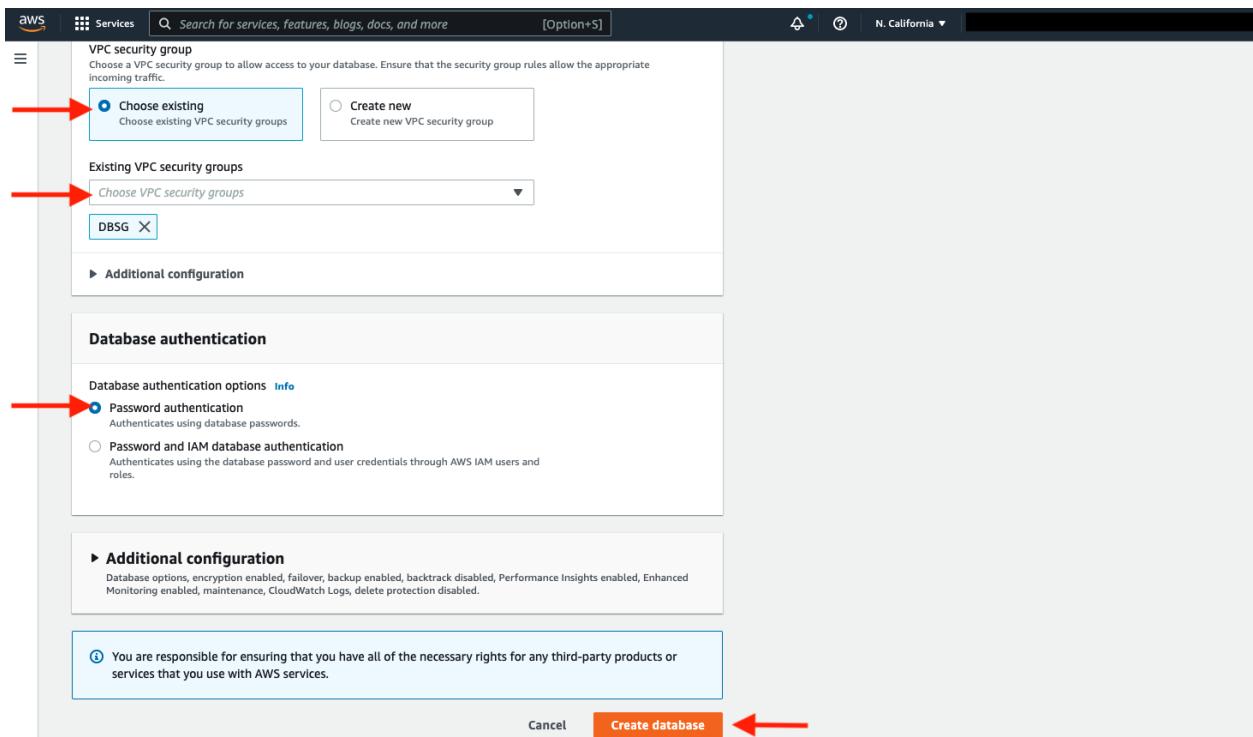
Under the **Templates** section choose **Dev/Test** since this isn't being used for production at the moment. Under **Settings** set a username and password of your choice and note them down since we'll be using password authentication to access our database.



Next, under **Availability and durability** change the option to create an Aurora Replica or reader node in a different availability zone. Under **Connectivity**, set the VPC, choose the subnet group we created earlier, and select no for public access.



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



3. When your database is provisioned, you should see a reader and writer instance in the database subnets of each availability zone. Note down the writer endpoint for your database for later use.

Amazon RDS

RDS > Databases > database-1

**Related**

DB identifier	Role	Engine	Region & AZ	Size	Status
database-1	Regional cluster	Aurora MySQL	us-west-1	2 instances	Available
database-1-instance-1	Writer instance	Aurora MySQL	us-west-1b	db.r5.large	Creating
database-1-instance-1-us-west-1c	Reader instance	Aurora MySQL	us-west-1c	db.r5.large	Creating

**Connectivity & security** | Monitoring | Logs & events | Configuration | Maintenance & backups | Tags

**Endpoints (2)**

Endpoint name	Status	Type	Port
database-1.cluster-cwbleayzsqgl.us-west-1.rds.amazonaws.com	Creating	Writer instance	3306
database-1.cluster-ro-cwbleayzsqgl.us-west-1.rds.amazonaws.com	Creating	Reader instance	3306

Manage IAM roles

## 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.

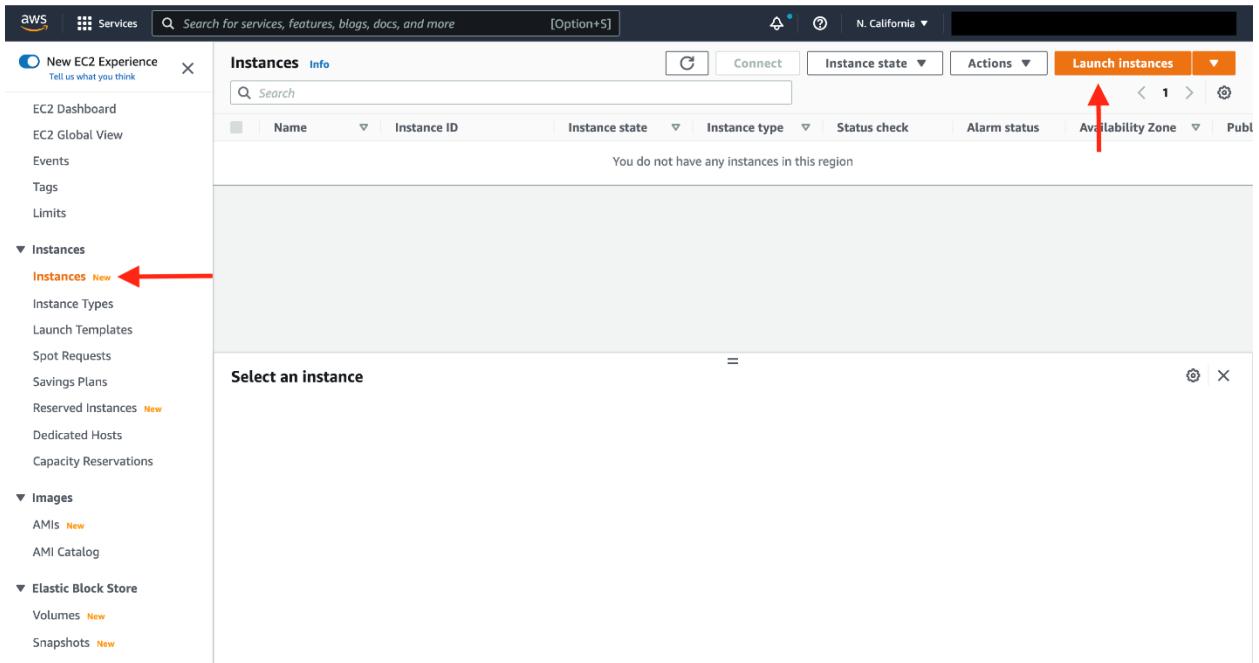
### Learning Objectives:

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

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

## # App Instance Deployment

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



## 2. Select the first Amazon Linux 2 AMI

**Step 1: Choose an Amazon Machine Image (AMI)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

**Quick Start**

- My AMIs
- AWS Marketplace
- Community AMIs
- Free tier only

AMI Name	Description	Select
Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type	- ami-0a8a24772b8f01294 (64-bit x86) / ami-0c4a6c5674a8bf60b (64-bit Arm)	<b>Select</b>
Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type	- ami-0577aef94c154720e (64-bit x86) / ami-04334a7eb2cf4a98f (64-bit Arm)	<b>Select</b>
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type	- ami-054965c6cd7c6e462 (64-bit x86) / ami-05f88a4bc91f4ea7 (64-bit Arm)	<b>Select</b>

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

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	<b>t2.micro</b> <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
<input type="checkbox"/>	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

Filter by: All instance families Current generation Show/Hide Columns

Cancel Previous Review and Launch **Next: Configure Instance Details**

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

Screenshot of the AWS EC2 instance creation wizard Step 3: Configure Instance Details.

The screenshot shows the configuration for a single instance (1) in the N. California region. The instance type is not specified. The network settings include a VPC (vpc-0d2a264d3ba772982) and a subnet (subnet-0f2b746756c354aaa). The IAM role selected is "aws-3tier-workshop-ec2-role". Other options like "Request Spot instances", "Auto-assign Public IP", and "DNS Hostname" are also visible.

Red arrows point to the Network, Subnet, and IAM role fields to highlight them.

Buttons at the bottom include Cancel, Previous, Review and Launch (highlighted in blue), and Next: Add Storage.

5. We'll be keeping the defaults for storage so click next twice. When you get to the tag screen input a **Name** as a key and call the instance AppLayer. It's a good idea to tag your instances so you can easily keep track of what each instance was created for. Click **Next: Configure Security Group**.

Screenshot of the AWS EC2 instance creation wizard Step 5: Add Tags.

The screenshot shows a tag being added with the key "Name" and value "AppLayer". The "Instances" checkbox is selected, indicating the tag will be applied to instances. The "Review and Launch" button is highlighted in blue, and the "Next: Configure Security Group" button is also highlighted with a red box.

- Earlier we created a security group for our private app layer instances, so go ahead and select that in this next section. Then click **Review and Launch**. Ignore the warning about connecting to port 22- we don't need to do that.

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:

- Create a new security group
- Select an existing security group ←

Security Group ID	Name	Description	Actions
sg-0049e953830bad2f4	DBSG	SG for our databases	<a href="#">Copy to new</a>
sg-0bf79a59048818994	default	default VPC security group	<a href="#">Copy to new</a>
sg-022b4036ffff83a40	internal-lb-sg	SG for the internal load balancer	<a href="#">Copy to new</a>
sg-072fe91349cba745f	internet-facing-lb-sg	External load balancer security group	<a href="#">Copy to new</a>
<b>sg-05bc5e33f3c5b34de</b> <span style="color:red">←</span>	<b>PrivateInstanceSG</b>	<b>SG for our private app tier sg</b>	<a href="#">Copy to new</a>
sg-0d115f758786138a5	WebTierSG	SG for the Web Tier	<a href="#">Copy to new</a>

Inbound rules for sg-05bc5e33f3c5b34de (Selected security groups: sg-05bc5e33f3c5b34de)

Type	Protocol	Port Range	Source	Description
Custom TCP Rule	TCP	4000	52.95.4.12/32	
Custom TCP Rule	TCP	4000	sg-022b4036ffff83a40 (internal-lb-sg)	

[Cancel](#) [Previous](#) **Review and Launch**  

- When you get to the **Review Instance Launch** page, review the details you configured and click **Launch**. You'll see a pop up about creating a key pair. Since we are using Systems Manager Session Manager to connect to the instance, **proceed without a keypair**. Click **Launch**.

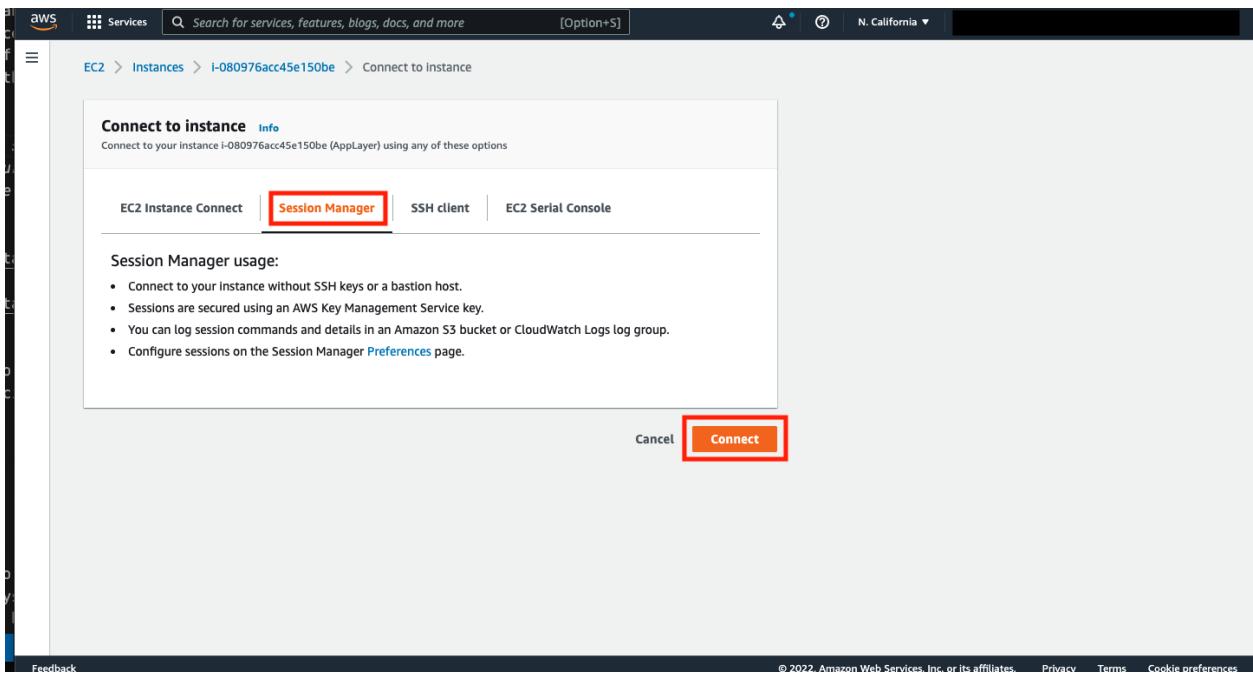
Screenshot of the AWS Step 7: Review Instance Launch page. The 'AMI Details' section shows 'Amazon Linux 2 AMI (HVM) - Kernel'. The 'Instance Type' section lists 't2.micro' with 1 vCPU. The 'Security Groups' section lists 'sg-05bc5e33f3c5b34de'. A red arrow points to the 'Proceed without a key pair' checkbox. Another red arrow points to the 'I acknowledge that without a key pair, I can connect to this instance only by using EC2 Instance Connect or if I know the password built into the AMI.' note. A third red arrow points to the 'Launch Instances' button, which is highlighted with a red box.

## 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.*

Screenshot of the AWS EC2 Instances page. The sidebar shows 'Instances New' (highlighted with a red arrow). The main table shows one instance named 'AppLayer' with ID 'i-08fe5fa3b830cccd3a', status 'Running', type 't2.micro', and availability zone 'us-west-1b'. A red arrow points to the 'Connect' button in the top right of the table header. Another red arrow points to the checkmark box next to the instance row. The bottom panel shows the details for instance 'i-08fe5fa3b830cccd3a (AppLayer)', including its summary and network information.



Project2 - Deploy a 3 Tier Archi | Editing AWS--3-Tier--web-Arch | AWS Three Tier Web Application | +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/connecttoinstance

aws workshop studio

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:

```
1 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:

```
1 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!*

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Type here to search Earnings upcoming 17:06 ENG 09-07-2024

Project2 - Deploy a 3 Tier Archi | Editing AWS--3-Tier--web-Arch | AWS Three Tier Web Application | +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configuredatabase

aws workshop studio

## Configure Database

1. Start by downloading the MySQL CLI:

```
1 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:

```
1 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.*

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Windows taskbar: Type here to search, Earnings upcoming, 17:06, 09-07-2024

Project2 - Deploy a 3 Tier Archi | Editing AWS--3-Tier--web-Arch | AWS Three Tier Web Application | +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configuredatabase

aws workshop studio

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

```
1 CREATE DATABASE webappdb;
```

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

```
1 SHOW DATABASES;
```

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

```
1 USE webappdb;
```

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

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Windows taskbar: Type here to search, Earnings upcoming, 17:06, 09-07-2024

The screenshot shows a browser window titled "catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configuredatabase". The page is a MySQL command-line interface. A sidebar on the left says "aws workshop studio". The main area has a heading "Then, create the following **transactions** table by executing this create table command:" followed by a code block:

```
1 CREATE TABLE IF NOT EXISTS transactions(id INT NOT NULL  
2 AUTO_INCREMENT, amount DECIMAL(10,2), description  
3 VARCHAR(100), PRIMARY KEY(id));
```

Below this, another heading says "Verify the table was created:" followed by a code block:

```
1 SHOW TABLES;
```

Further down, a heading says "5. Insert data into table for use/testing later:" followed by a code block:

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

The bottom of the browser window shows standard Windows taskbar icons and system status.

The screenshot shows a browser window titled "catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configuredatabase". The page is a MySQL command-line interface. A sidebar on the left says "aws workshop studio". The main area has a heading "5. Insert data into table for use/testing later:" followed by a code block:

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

Below this, a heading says "Verify that your data was added by executing the following command:" followed by a code block:

```
1 SELECT * FROM transactions;
```

Further down, a heading says "6. When finished, just type exit and hit enter to exit the MySQL client." Below this are "Previous" and "Next" buttons. The bottom of the browser window shows standard Windows taskbar icons and system status.

## Configure App Instance

1. 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 `webapppdb` 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 with the 'Objects' tab selected. The bucket name is 'awsthreetierworkshop'. The 'Upload' button is highlighted with a red box. The table below shows 0 objects.

Name	Type	Last modified	Size	Storage class
No objects				

The screenshot shows the 'Upload' dialog in the AWS S3 console. The 'Files and folders' section shows 6 files and 47.4 KB total size. The 'Add folder' button is highlighted with a red arrow. The 'Upload' button at the bottom right is also highlighted with a red arrow.

Name	Folder	Type	Size
DbConfig.js	app-tier/	text/javascript	190.0 B
README.md	app-tier/	-	12.0 B
TransactionService.js	app-tier/	text/javascript	1.7 KB
index.js	app-tier/	text/javascript	3.1 KB
package-lock.json	app-tier/	application/json	41.8 KB
package.json	app-tier/	application/json	655.0 B

Project2 - Deploy a 3 Tier Arch | Editing AWS--3-Tier--web-Arch | AWS Three Tier Web Application

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configureinstance

aws workshop studio

2. Upload the **app-tier** folder to the S3 bucket that you created in part 0.

3. 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).

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

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

```
1 nvm install 16
2 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.

```
1 npm install -g pm2
```

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Project2 - Deploy a 3 Tier Arch! | Editing AWS--3-Tier--web-Arch | AWS Three Tier Web Application | +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configureinstance

aws workshop studio

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:

```
1 cd ~/  
2 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.

```
1 cd ~/app-tier  
2 npm install  
3 pm2 start index.js
```

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

```
1 pm2 list
```

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

The screenshot shows a browser window titled "AWS Three Tier Web Application" with the URL "catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configureinstance". The page displays a terminal-like interface with the command "1 pm2 logs". Below the terminal, a note says: "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." A section titled "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:" contains the command "1 pm2 startup". The Windows taskbar at the bottom shows various pinned icons and the date/time as 09-07-2024.

The screenshot shows a browser window titled "AWS Three Tier Web Application" with the URL "catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part3/configureinstance". The page displays a terminal-like interface with the command "[PM2] To setup the Startup Script, copy/paste the following command: sudo env PATH=\$PATH:/home/ec2-user/.nvm/versions/node/v16.0.0/bin /home/ec2-u". Below the terminal, a note says: "After running this you will see a message similar to this." A section titled "DO NOT run the above command, rather you should copy and past the command in the output you see in your own terminal. After you run it, save the current list of node processes with the following command:" contains the command "1 pm2 save". At the bottom right are "Previous" and "Next" buttons. The Windows taskbar at the bottom shows various pinned icons and the date/time as 09-07-2024.

The screenshot shows the AWS Workshop Studio interface with the title "Test App Tier". It contains instructions to run a health check endpoint and a code snippet for curl. Below it, a success message is shown. A note says to test the database connection by hitting a local endpoint.

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.

```
1 curl http://localhost:4000/health
```

The response should looks like the following:

```
1 This is the health check*
```

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

The screenshot shows the AWS Workshop Studio interface with the title "Test App Tier". It contains instructions to run a transaction endpoint and a code snippet for curl. Below it, a note says to see the response containing test data.

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

```
1 curl http://localhost:4000/transaction
```

You should see a response containing the test data we added earlier:

```
1 {"result": [{"id": 1, "amount": 400, "description": "groceries"}, {"id": 2, "amount": 100, "description": "class"}, {"id": 3, "amount": 200, "description": "books"}]}
```

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.

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

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

## 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**.

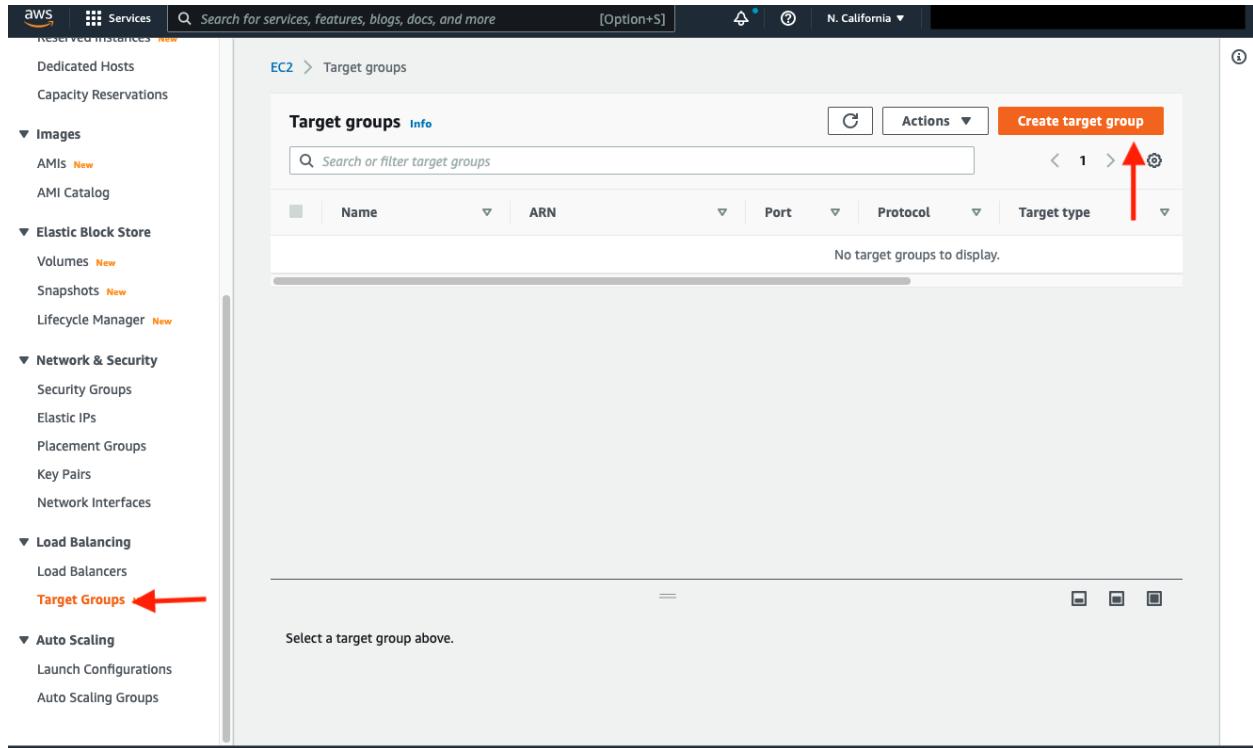
The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation pane with various EC2-related options. The main area displays a table of instances. A context menu is open over the first instance, which has a blue header bar. The menu items include 'Create image', 'Image and templates', and 'Launch more like this'. Red arrows highlight the 'Create image' and 'Image and templates' buttons.

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' wizard. It has several sections: 'Instance ID' (i-080976acc45e150be), 'Image name' (AppTierImage), 'Image description - optional' (App Tier), 'No reboot' (checkbox), 'Instance volumes' (EBS volume setup), 'Tags - optional' (radio buttons for 'Tag image and snapshots together' or 'Tag image and snapshots separately'), and 'Add tag' (button). At the bottom right is a large red arrow pointing to the 'Create image' button.

## 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**



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.

Screenshot of the AWS CloudFront console showing the 'Basic configuration' step for creating a target group.

**Step 2** Register targets

**Basic configuration**

Settings in this section cannot be changed after the target group is created.

**Choose a target type**

- Instances
  - Supports load balancing to instances within a specific VPC.
  - Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.
- IP addresses
  - Supports load balancing to VPC and on-premises resources.
  - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
  - Offers flexibility with microservice based architectures, simplifying inter-application communication.
  - Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
- Lambda function
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.
- Application Load Balancer
  - Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
  - Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

**Target group name**

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

**Protocol**  **Port**

hen, 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**.

Screenshot of the AWS CloudFront console showing the 'Protocol and VPC' step for creating a target group.

**Protocol**  **Port**

**VPC**  
Select the VPC with the instances that you want to include in the target group.

vpc-0d2a264d3ba772982  
IPv4: 10.0.0.0/16

**Protocol version**

- HTTP1**  
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
- HTTP2**  
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
- gRPC**  
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

**Health checks**

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

**Health check protocol**

**Health check path**  
Use the default path of "/" to ping the root, or specify a custom path if preferred.

Up to 1024 characters allowed.

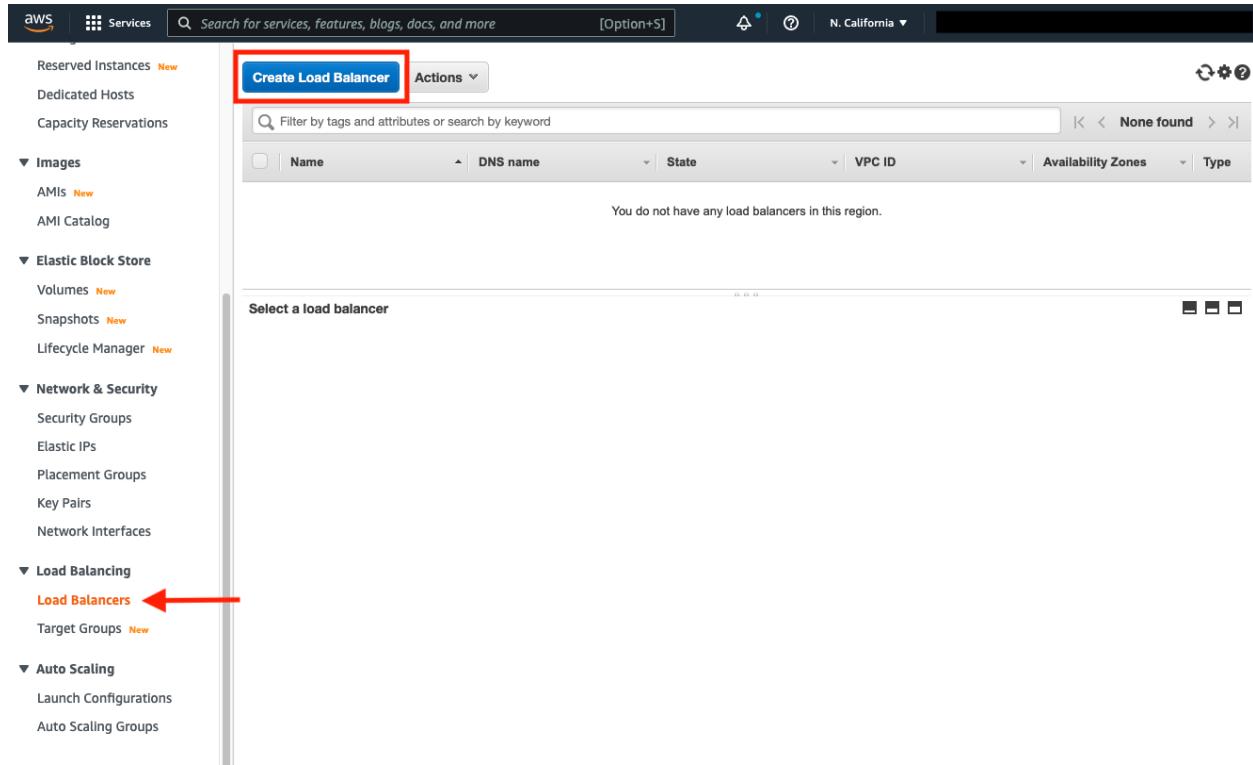
**Advanced health check settings**

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

The screenshot shows the AWS EC2 Target Groups interface. At the top, there is a search bar and a navigation bar with 'Services' and 'N. California'. Below the search bar is a table header with columns: Instance ID, Name, State, Security groups, Zone, and Subnet ID. A single instance is listed: 'i-080976acc45e150be' (AppLayer, running, PrivateInstanceSG, us-west-1b, subnet-0f2b746756c354aaa). Below the table, it says '0 selected'. Under 'Ports for the selected instances', the port '4000' is specified. There is also a note: '1-65535 (separate multiple ports with commas)'. A button 'Include as pending below' is present. The main area is titled 'Review targets' and contains a section for 'Targets (0)'. A red arrow points to the 'Targets (0)' heading. Below it is a table with columns: All, Remove, Health status, Instance ID, Name, Port, State, Security groups, Zone, and Subnet ID. It displays the message 'No instances added yet' and a note: '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 a red-bordered 'Create target group' button.

## Internal Load Balancer

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



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

Screenshot of the AWS Load Balancer types page.

**Load balancer types**

Application Load Balancer <a href="#">Info</a>	Network Load Balancer <a href="#">Info</a>	Gateway Load Balancer <a href="#">Info</a>
<p>Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.</p> <p><a href="#">Create</a></p>	<p>Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.</p> <p><a href="#">Create</a></p>	<p>Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENVE. These appliances enable you to improve security, compliance, and policy controls.</p> <p><a href="#">Create</a></p>

- 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.

Screenshot of the Create Application Load Balancer page.

**Create Application Load Balancer [Info](#)**

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

**How Application Load Balancers work**

**Basic configuration**

**Load balancer name**  
Name must be unique within your AWS account and cannot be changed after the load balancer is created.  
**app-tier-internal-lb**

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

**Scheme [Info](#)**  
Scheme cannot 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 subnet

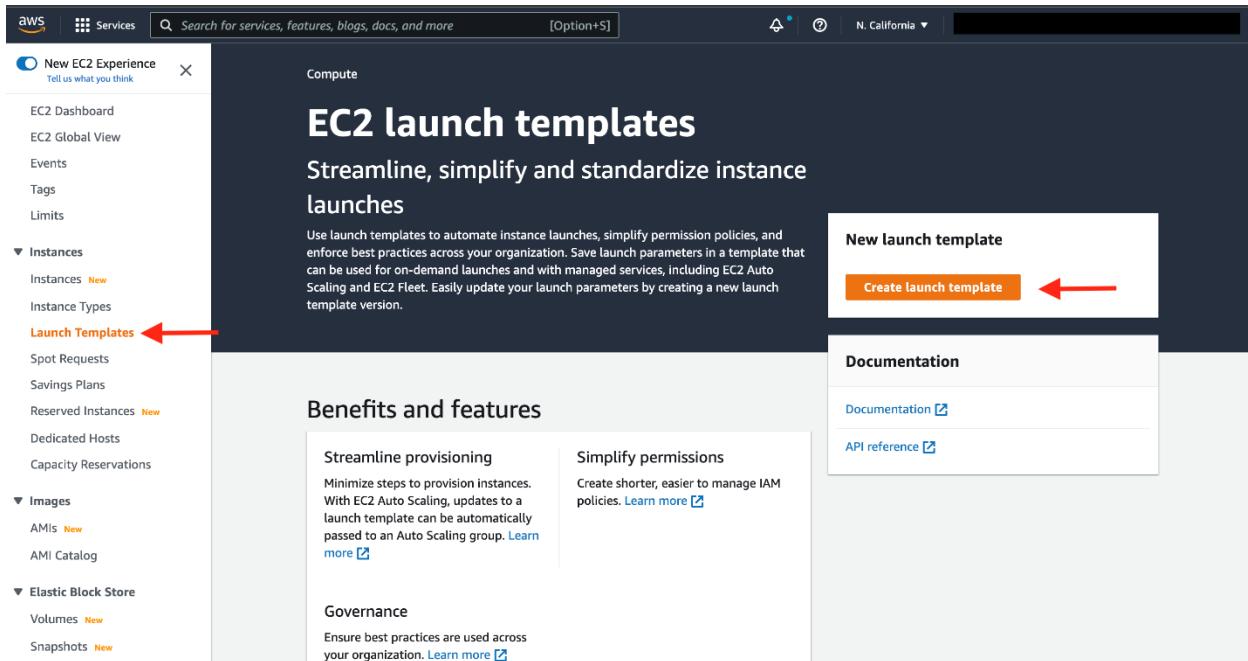
The screenshot shows the AWS VPC configuration page. In the 'VPC Info' section, the VPC 'awsthreetierworkshop' is selected, indicated by a red arrow. In the 'Mappings' section, two Availability Zones are listed: 'us-west-1b' and 'us-west-1c'. Under 'us-west-1b', a subnet 'subnet-0f2b746756c354aaa' is selected for 'Private-Subnet-AZ1', indicated by a red arrow. Under 'us-west-1c', a subnet 'subnet-01a29120b002287a7' is selected for 'Private-Subnet-AZ2', also indicated by a red arrow.

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 AWS Security Groups configuration page. In the 'Security groups' section, a security group 'internal-lb-sg sg-022b4036ffff83a40' is selected, indicated by a red arrow. In the 'Listeners and routing' section, a new listener 'Listener HTTP:80' is being configured. The 'Protocol' is set to 'HTTP' and the 'Port' is set to '80', indicated by a red arrow. The 'Default action' dropdown is set to 'Forward to AppTierTargetGroup', indicated by another red arrow. A 'Create target group' button is visible below the dropdown.

# Launch Template

1. 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**.



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

# 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.

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 AWS EC2 'Create launch template' wizard. In the 'Firewall (security groups)' section, there are two buttons: 'Select existing security group' (with a red arrow pointing to it) and 'Create security group'. Below these are dropdown menus for 'Select security groups' containing 'PrivateInstanceSG sg-05bc5e33f3c5b34de' and 'VPC: vpc-0d2a264d3ba772982'. A 'Compare security group rules' link is also present. The 'Configure storage' section shows '1x 8 GiB gp2' selected for the root volume. A note indicates free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. The 'Resource tags' section notes that no tags are currently included. On the right, the 'Summary' panel shows the selected AMI (AppTierAMI), instance type (t2.micro), and storage (1 volume(s) - 8 GiB). A tooltip for the free tier is displayed, and at the bottom are 'Cancel' and 'Create launch template' buttons.

The screenshot shows the 'Advanced details' section of the EC2 launch template creation wizard. It includes fields for 'Purchasing option' (checkbox for Request Spot Instances), 'IAM instance profile' (dropdown set to 'aws-3tier-workshop-ec2-role' with a red arrow pointing to it), 'Hostname type' (dropdown set to 'Don't include in launch template'), 'DNS Hostname' (checkboxes for IPv4 and IPv6 DNS requests), and 'Shutdown behavior' (dropdown set to 'Don't include in launch template'). On the right, the 'Summary' panel shows the selected AMI (App Tier), instance type (t2.micro), and storage (1 volume(s) - 8 GiB). A tooltip for the free tier is displayed, and at the bottom are 'Cancel' and 'Create launch template' buttons, with the 'Create launch template' button highlighted by a red box.

## 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**.

Screenshot of the AWS EC2 Auto Scaling homepage. The left sidebar shows navigation options like Capacity Reservations, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. Under Auto Scaling, the 'Auto Scaling Groups' option is highlighted with a red arrow. The main content area features a large title 'Amazon EC2 Auto Scaling' and a sub-section 'How it works' with a diagram showing an 'Auto Scaling group' containing four instances. A call-to-action button 'Create Auto Scaling group' is prominently displayed.

Screenshot of the AWS EC2 Auto Scaling homepage, identical to the first one but with a different red arrow pointing to the 'Auto Scaling Groups' link in the sidebar under the 'Auto Scaling' section.

## 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 homepage. On the left, there's a sidebar with various navigation options like Capacity Reservations, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. Under Auto Scaling, the 'Auto Scaling Groups' link is highlighted with a red arrow. The main content area features a large heading 'Amazon EC2 Auto Scaling' with the subtext 'helps maintain the availability of your applications'. Below this is a description of what Auto Scaling groups are and how they work, accompanied by a diagram showing an 'Auto Scaling group' containing several EC2 instances. To the right, there's a 'Create Auto Scaling group' button and a 'Pricing' section.

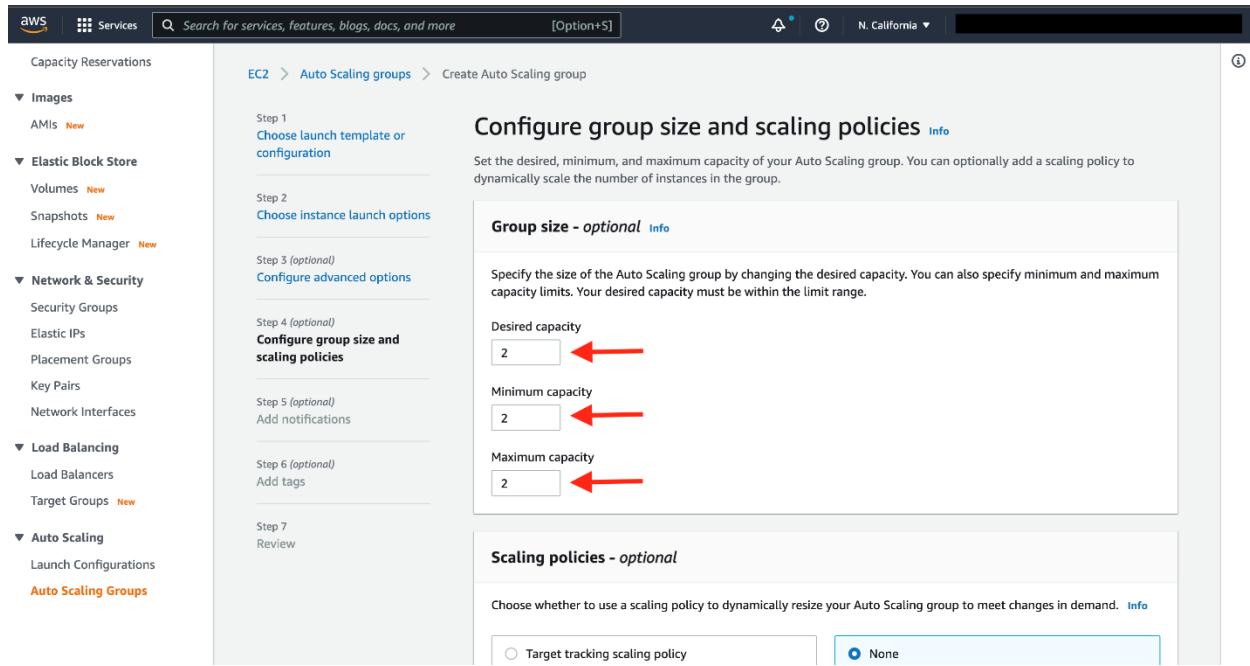
2. Give your Auto Scaling group a name, and then select the Launch Template we just created and click next.

This screenshot shows the 'Choose launch template or configuration' step in the AWS Auto Scaling group creation wizard. The left sidebar shows the same navigation as the previous screenshot. The main form has a series of steps on the left: Step 1 (Choose launch template or configuration), Step 2 (Choose instance launch options), Step 3 (optional) Configure advanced options, Step 4 (optional) Configure group size and scaling policies, Step 5 (optional) Add notifications, Step 6 (optional) Add tags, and Step 7 (Review). Step 1 is currently active. In the 'Name' section, a red arrow points to the input field where 'AppTierASG' is typed. Another red arrow points to the 'Launch template' dropdown, which shows 'AppTierLaunchTemplate' selected. The 'Switch to launch configuration' link is also visible.

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.

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.

- 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.



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.*

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

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

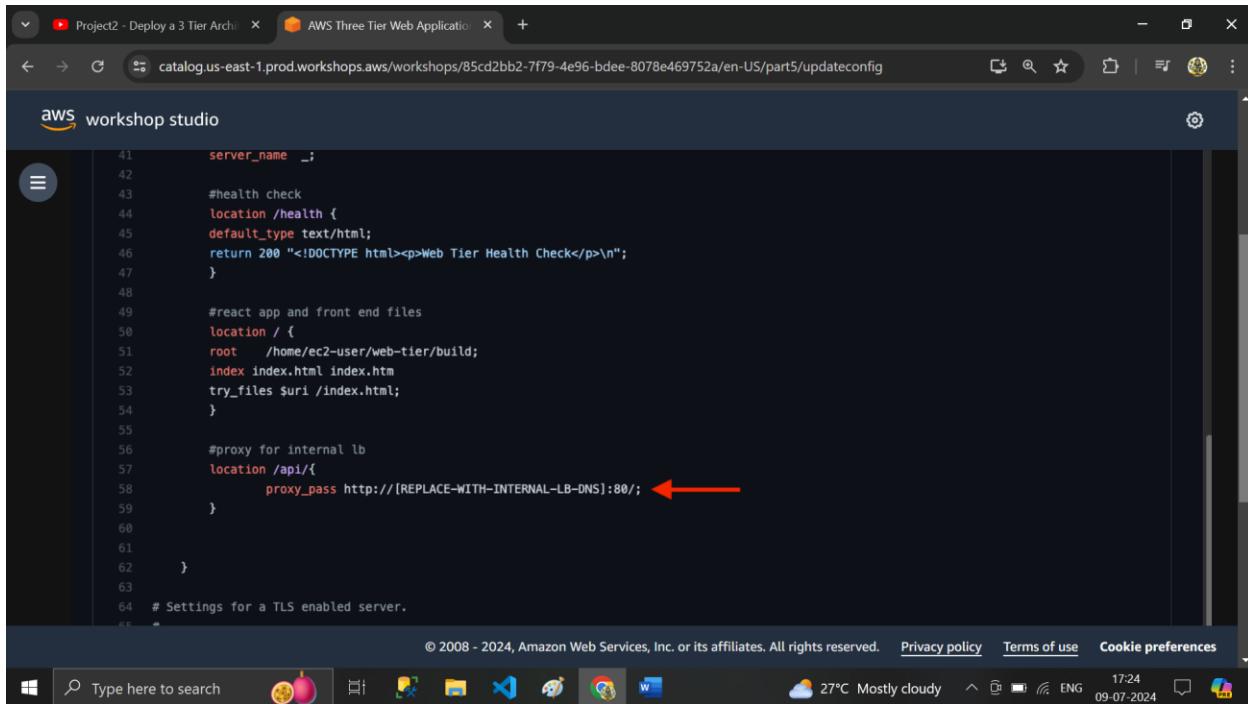
27°C Mostly cloudy 17:24 09-07-2024

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.

```
40     listen      [::]:80;
41     server_name _;
42
43     #health check
44     location /health {
45         default_type text/html;
46         return 200 "<!DOCTYPE html><p>Web Tier Health Check</p>\n";
47     }
48
49     #react app and front end files
50     location / {
51         root    /home/ec2-user/web-tier/build;
52         index  index.html index.htm;
53         try_files $uri /index.html;
54     }
```

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

27°C Mostly cloudy 17:24 09-07-2024



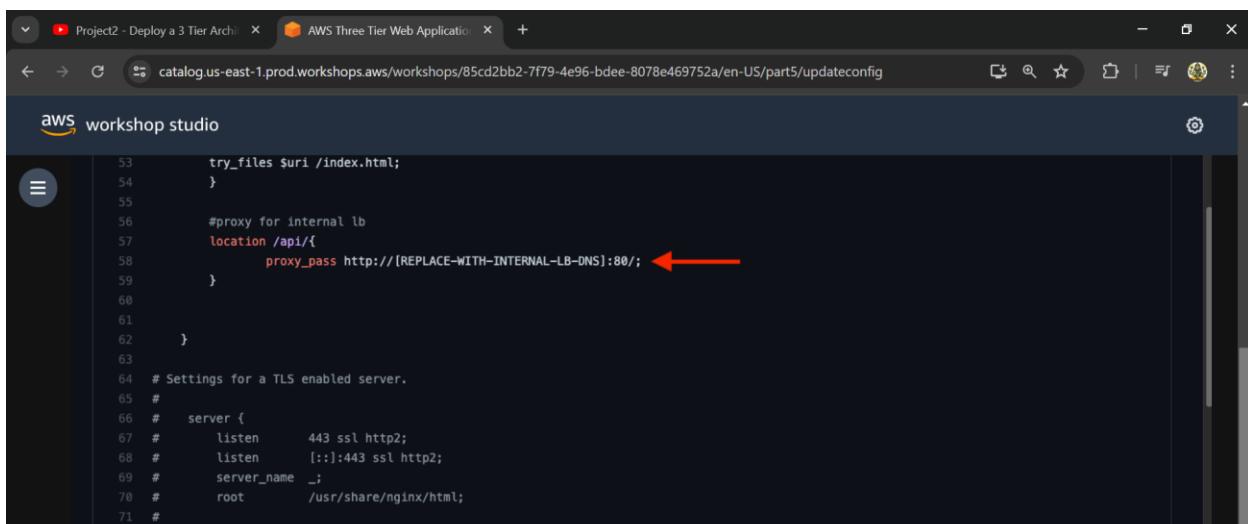
```
aws workshop studio

41     server_name _;
42
43     #health check
44     location /health {
45         default_type text/html;
46         return 200 "<!DOCTYPE html><p>Web Tier Health Check</p>\n";
47     }
48
49     #react app and front end files
50     location / {
51         root    /home/ec2-user/web-tier/build;
52         index  index.html index.htm;
53         try_files $uri /index.html;
54     }
55
56     #proxy for internal lb
57     location /api/ {
58         proxy_pass http://[REPLACE-WITH-INTERNAL-LB-DNS]:80/; ←
59     }
60
61     }
62
63     # Settings for a TLS enabled server.
64
65 "
```

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Type here to search

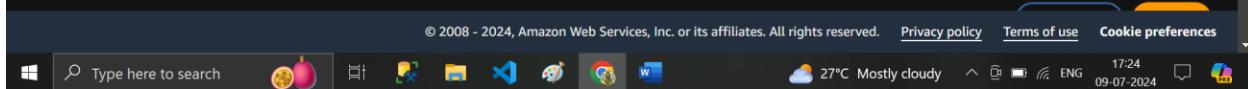
27°C Mostly cloudy 17:24 09-07-2024



```
aws workshop studio

53     try_files $uri /index.html;
54
55
56     #proxy for internal lb
57     location /api/ {
58         proxy_pass http://[REPLACE-WITH-INTERNAL-LB-DNS]:80/; ←
59     }
60
61     }
62
63     # Settings for a TLS enabled server.
64
65     #
66     #   server {
67     #       listen      443 ssl http2;
68     #       listen      [::]:443 ssl http2;
69     #       server_name _;
70     #       root        /usr/share/nginx/html;
71     #   }
```

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



Project2 - Deploy a 3 Tier Arch | AWS Three Tier Web Application

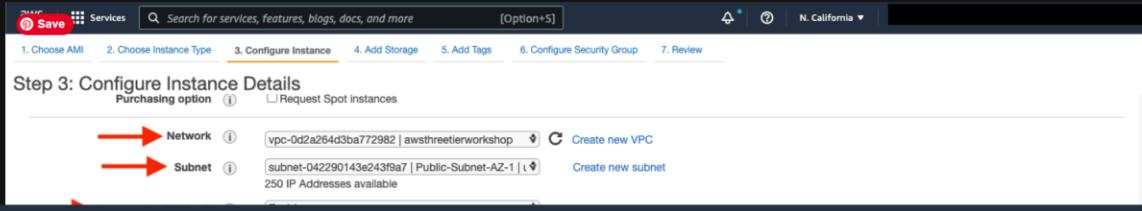
catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part5/createinstance

aws workshop studio

AWS Three Tier Web Architecture > Part 5: Web Tier Instance Deployment > Web Instance Deployment

## Web Instance Deployment

1. 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.



aws Services Search for services, features, blogs, docs, and more [Option+S]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 3: Configure Instance Details

Purchasing option  Request Spot instances

Network: vpc-0d2a264d3ba772982 | awsthreetierworkshop  Subnet: subnet-042290143e243f9a7 | Public-Subnet-AZ-1  250 IP Addresses available

Auto-assign Public IP:

Hostname type: Use subnet setting (IP name)  Enable IP name IPv4 (A record) DNS requests  Enable resource-based IPv4 (A record) DNS requests  Enable resource-based IPv6 (AAAA record) DNS requests

Placement group:  Add instance to placement group Capacity Reservation: Open

Domain join directory: No directory

IAM role: aws-3tier-workshop-ec2-role

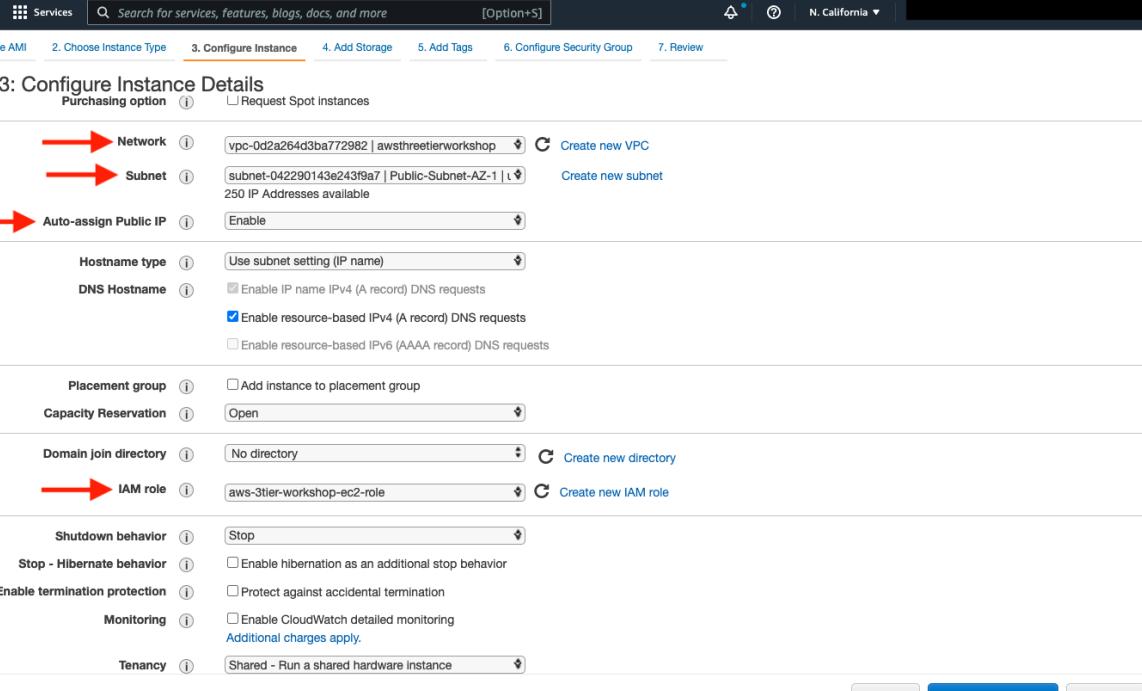
Shutdown behavior: Stop  Enable hibernation as an additional stop behavior

Stop - Hibernate behavior:  Protect against accidental termination

Enable termination protection:  Enable CloudWatch detailed monitoring Additional charges apply.

Monitoring: Shared - Run a shared hardware instance

Cancel Previous Review and Launch Next: Add Storage



Screenshot of the AWS EC2 instance creation process, Step 6: Configure Security Group.

The page shows a list of existing security groups:

Security Group ID	Name	Description	Actions
sg-0049e953830bad2f4	DBSG	SG for our databases	<a href="#">Copy to new</a>
sg-0bf79a59048818994	default	default VPC security group	<a href="#">Copy to new</a>
sg-022bd036ff8f3a40	Internal-lb-sg	SG for the internal load balancer	<a href="#">Copy to new</a>
sg-072fe91349cba745f	internet-facing-lb-sg	External load balancer security group	<a href="#">Copy to new</a>
sg-05bc5e33fc5b34de	PrivateInstanceSG	SG for our private app tier sg	<a href="#">Copy to new</a>
<b>sg-0d115f758786138a5</b>	<b>WebTierSG</b>	SG for the Web Tier	<a href="#">Copy to new</a>

A red arrow points to the selected security group "sg-0d115f758786138a5".

Inbound rules for sg-0d115f758786138a5 (Selected security groups: sg-0d115f758786138a5)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	67.176.251.60/32	
HTTP	TCP	80	sg-072fe91349cba745f (internet-facing-lb-sg)	

Buttons at the bottom: Cancel, Previous, Review and Launch (highlighted).

Then at the end, proceed without a key pair for this instance.

Screenshot of the AWS Workshop Studio interface showing the "Connect to Instance" step.

The URL in the browser is catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part5/connecttoinstance

The page title is AWS Three Tier Web Architecture > Part 5: Web Tier Instance Deployment > Connect to Instance

## Connect to Instance

1. 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:

```
1 sudo -su ec2-user
2 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.

Buttons at the bottom: Previous, Next (highlighted).

Page footer: © 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy policy Terms of use Cookie preferences

The screenshot shows a browser window titled "AWS Three Tier Web Application" with the URL "catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part5/configureinstance". The page is titled "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 :

```
1 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
2 source ~/.bashrc
3 nvm install 16
4 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:

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy policy Terms of use Cookie preferences

The screenshot shows a browser window titled "AWS Three Tier Web Application" with the URL "catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part5/configureinstance". The page is titled "Configure Web Instance".

1 cd ~/web-tier
2 npm install
3 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.

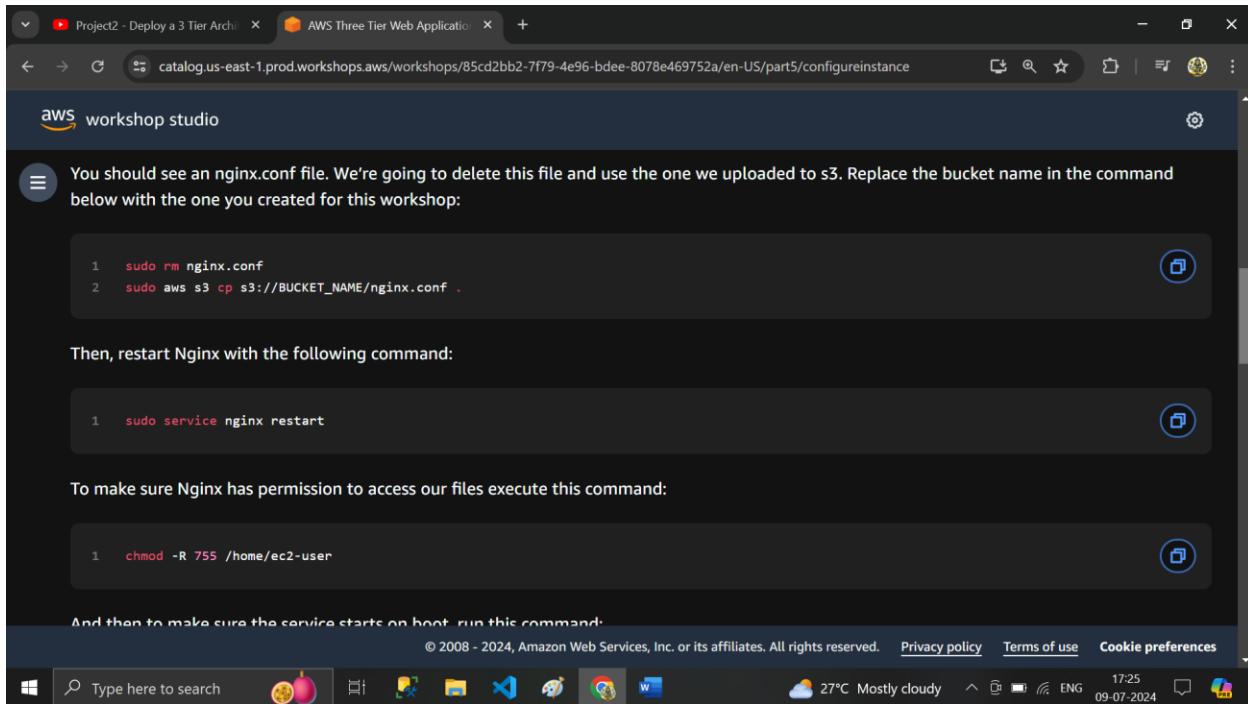
```
1 sudo amazon-linux-extras install nginx1 -y
```

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:

```
1 cd /etc/nginx
2 ls
```

You should see an nginx.conf file. We're going to delete 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:

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy policy Terms of use Cookie preferences



You should see an nginx.conf file. We're going to delete 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:

```
1 sudo rm nginx.conf
2 sudo aws s3 cp s3://BUCKET_NAME/nginx.conf .
```

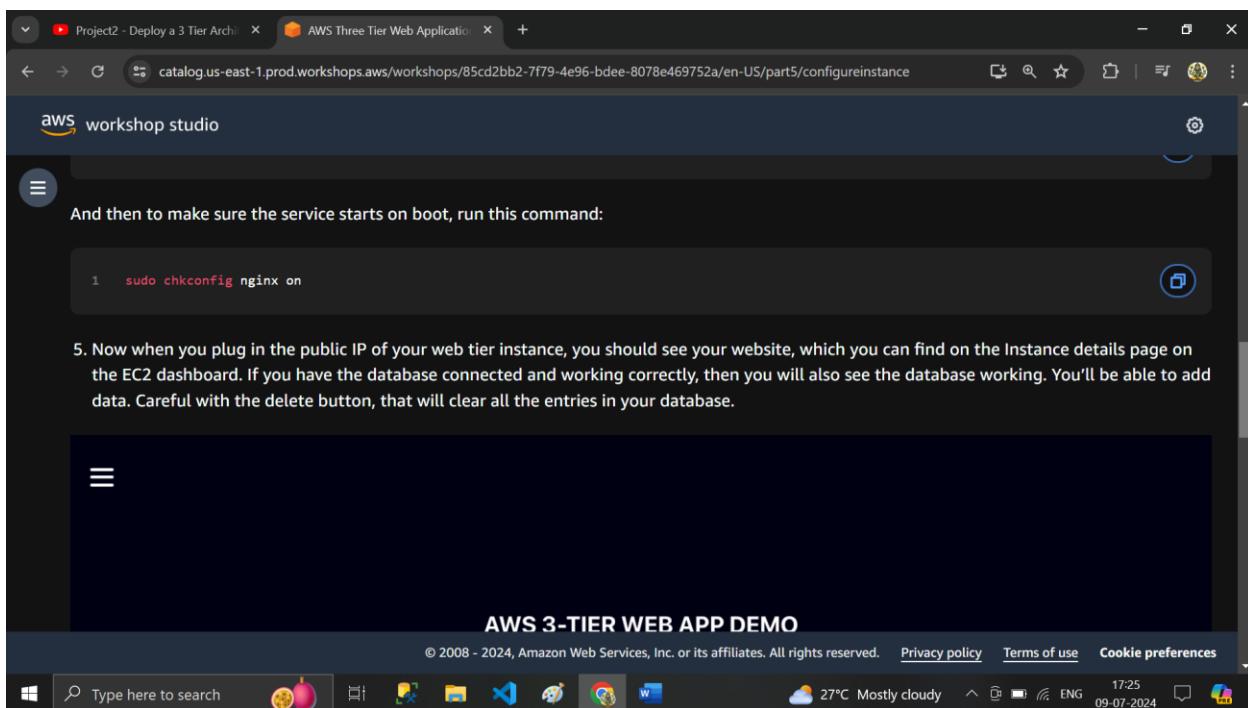
Then, restart Nginx with the following command:

```
1 sudo service nginx restart
```

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

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

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

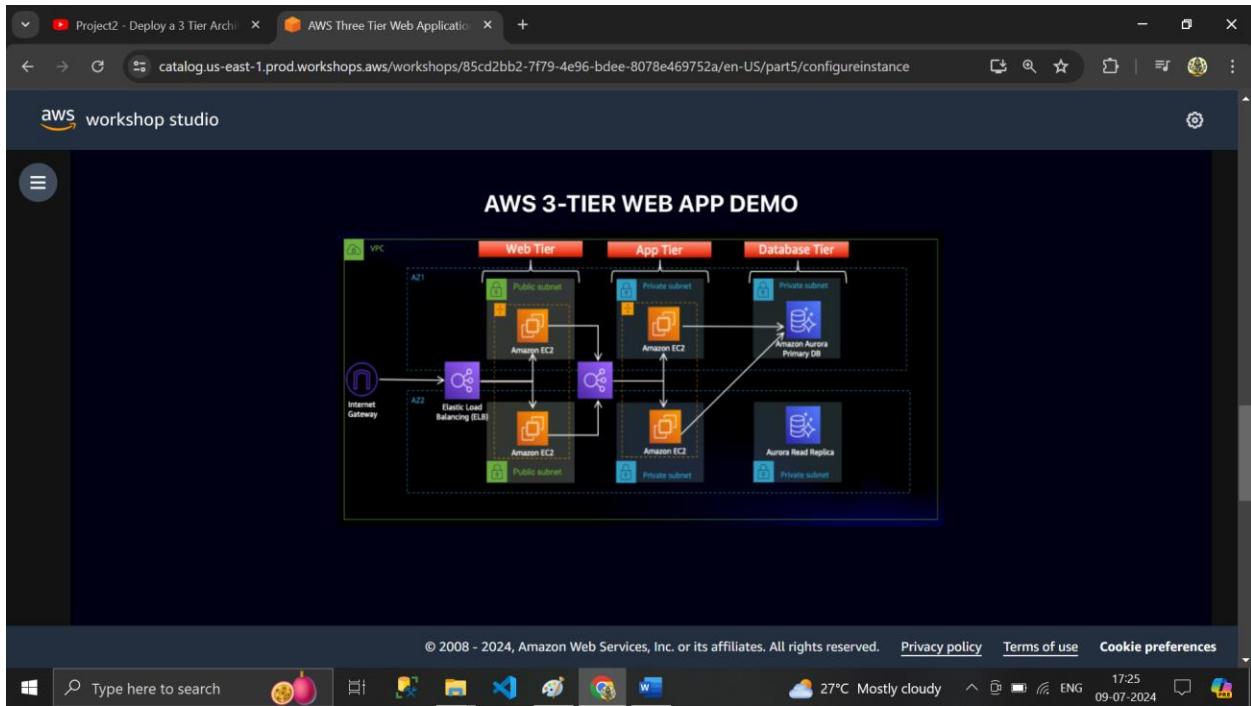


And then to make sure the service starts on boot, run this command:

```
1 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.

AWS 3-TIER WEB APP DEMO



Project2 - Deploy a 3 Tier Archi

AWS Three Tier Web Application

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part5/configureinstance

aws workshop studio

### AURORA DATABASE DEMO PAGE

**HOME**

**DB DEMO** ←

ID	AMOUNT	DESC
ADD		
1	400	groceries

DEL

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Windows Type here to search 27°C Mostly cloudy 17:25 09-07-2024

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

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

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**.

Instances (1/4) **Create Image**

Name	Instance ID	Instance state	Instance type	Status
–	i-08154197388ed6033	Running	t2.micro	2/2
AppLayer	i-080976acc45e150be	Running	t2.micro	2/2
<b>WebTier</b>	<b>i-0c9e079coa9646802</b>	<b>Running</b>		

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with various options like EC2 Dashboard, Instances, and Images. Under Instances, 'Instances' is highlighted with a red arrow. The main area shows a table of instances with columns for Name, Instance ID, Instance state, Instance type, and Status. Three instances are listed: 'WebTier' (selected), 'AppLayer', and another unnamed instance. A context menu is open over the 'WebTier' row, with a red arrow pointing to the 'Create image' option. The menu also includes 'Create template from instance' and 'Launch more like this'. To the right of the table, there are buttons for Connect, Instance state, Actions, and Launch Instances.

The screenshot shows the 'Create image' dialog box. At the top, it says 'Create image' with a 'Info' link. Below that, a note states: '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 configuration of an existing instance.' The 'Instance ID' field is populated with 'i-0c9e079c0a9646802 (WebTier)'. The 'Image name' field has 'WebTierimage' entered, with a red arrow pointing to it. The 'Image description - optional' field contains 'Image of our Web tier Instance', also with a red arrow pointing to it. There are checkboxes for 'No reboot' and 'Enable'. The 'Instance volumes' section shows one EBS volume (/dev/xvda) with a size of 8 GiB, an IOPS of 100, and an encrypted status. The dialog also includes 'Create new snapshot from this volume' and 'Add volume' buttons. At the bottom, there are links for 'Privacy policy', 'Terms of use', and 'Cookie preferences'.

**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 configuration of an existing instance.

Instance ID  
i-0c9e079c9a9646802 (WebTier)

Image name  
 ←

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

Image description - optional  
 ←

Maximum 255 characters

No reboot  
 Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/x...	Create new snapshot fr...	8	EBS General Purpose S...	100		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

[Add volume](#)

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional

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.

Tag image and snapshots together  
Tag the image and the snapshots with the same tag.

Tag image and snapshots separately  
Tag the image and the snapshots with different tags.

No tags associated with the resource.

[Add tag](#)

You can add 50 more tags.

[Create image](#) ←

Project2 - Deploy a 3 Tier Archi... AWS Three Tier Web Application +

catalog.us-east-1.prod.workshops.aws/workshops/85cd2bb2-7f79-4e96-bdee-8078e469752a/en-US/part6/targetgroup

aws workshop studio

AWS Three Tier Web Architecture > Part 6: External Load Balancer and Auto Scaling > Target Group

## 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**.

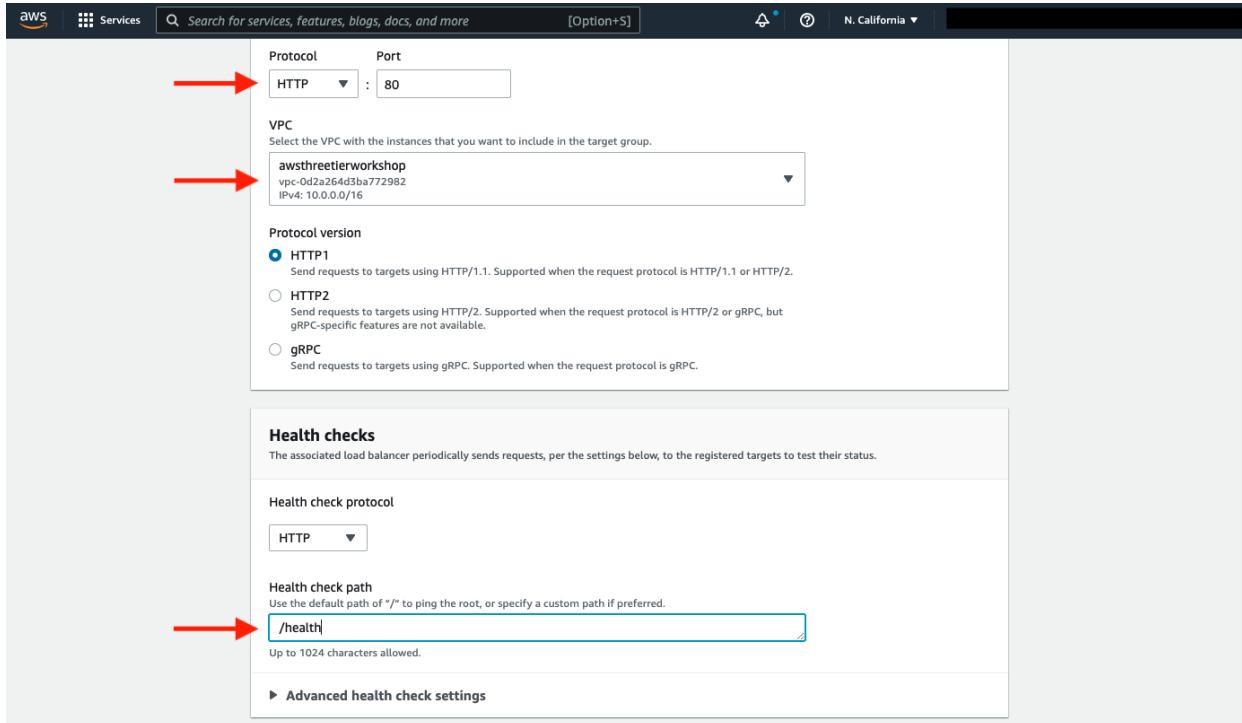
The screenshot shows the AWS EC2 Target Groups page. On the left sidebar, under 'Images', there is a 'AMIs' section. The main area displays a table for 'Target groups (1)'. The first row shows a target group named 'AppTierTargetGroup' with ARN 'arn:aws:elasticloadbalancing...', port '4000', protocol 'HTTP', target type 'Instance', and load balancer 'app-tier-internal-lb'. At the top right of the table, there is a red arrow pointing to the 'Create target group' button.

The screenshot shows the AWS EC2 Target groups page. On the left, there's a navigation sidebar with various services like Savings Plans, Reserved Instances, AMIs, and Auto Scaling. Under the Load Balancing section, 'Target Groups' is highlighted with a red arrow. The main content area shows a table titled 'Target groups (1) Info'. The table has columns for Name, ARN, Port, Protocol, Target type, and Load balancer. One row is selected, showing 'AppTierTargetGroup' as the name, 'arn:aws:elasticloadbalancing:...', port 4000, protocol HTTP, target type Instance, and load balancer app-tier-internal-lb. Above the table is a search bar and an 'Actions' dropdown. A prominent orange button at the top right says 'Create target group'. A red arrow points from the 'Target Groups' link in the sidebar to the 'Create target group' button.

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

The screenshot shows the 'Specify group details' step of the 'Create target group' wizard. It's divided into two tabs: 'Step 1 Specify group details' (selected) and 'Step 2 Register targets'. The 'Basic configuration' section contains a note about settings being immutable after creation. Below is a 'Choose a target type' section with four options: 'Instances' (selected), 'IP addresses', 'Lambda function', and 'Application Load Balancer'. A red arrow points to the 'Instances' option. At the bottom is a 'Target group name' input field containing 'WebTierTargetGroup', with another red arrow pointing to it.

Then, set the protocol to **HTTP** and the port to 80. Remember this is the port NGINX is listening on. Select the VPC we've been using thus far, and then change the health check path to be **/health**. Click **Next**.



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

aws Services Search for services, features, blogs, docs, and more [Option+S] N. California

<input type="checkbox"/>	i-0c9e079c0a9646802	WebTier	running	WebTierSG	us-west-1b	subnet-042290143e243f9a7
<input type="checkbox"/>	i-0ef244a9e26b075d3		running	PrivateInstanceSG	us-west-1b	subnet-0f2b746756c354aaa

0 selected

Ports for the selected Instances  
Ports for routing traffic to the selected instances.

80  
1-65535 (separate multiple ports with commas)

Include as pending below

Review targets

Targets (0)

All Filter resources by property or value

Remove Health status Instance ID Name Port State Security groups Zone Subnet ID

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

## Internet Facing Load Balancer

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

aws Services Search for services, features, blogs, docs, and more [Option+S] N. California

Reserved Instances New  
Dedicated Hosts  
Capacity Reservations

Images AMIs New  
AMI Catalog

Elastic Block Store Volumes New  
Snapshots New  
Lifecycle Manager New

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

Load Balancing Load Balancers **New** ←  
Target Groups New

Auto Scaling Launch Configurations  
Auto Scaling Groups

Create Load Balancer Actions

Filter by tags and attributes or search by keyword

Name	DNS name	State	VPC ID	Availability Zones	Type
app-tier-internal-lb	internal-app-tier-internal-lb-2...	Active	vpc-0d2a264d3ba772982	us-west-1c, us-west-1b	application

Load balancer: app-tier-internal-lb

Description Listeners Monitoring Integrated services Tags

Basic Configuration

Name	app-tier-internal-lb
ARN	arn:aws:elasticloadbalancing:us-west-1:192462669998:loadbalancer/app/app-tier-internal-lb/14c8ab57576fb12
DNS name	internal-app-tier-internal-lb-289891698.us-west-1.elb.amazonaws.com (A Record)
State	Active
Type	application
Scheme	internal
IP address type	Ipv4
<span style="border: 1px solid #ccc; padding: 2px 10px;">Edit IP address type</span>	
VPC	vpc-0d2a264d3ba772982
Availability Zones	subnet-01a29120b002287a7 - us-west-1c IPv4 address: Assigned from CIDR 10.0.4.0/24

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

The screenshot shows the 'Load balancer types' section of the AWS Management Console. It compares three types of load balancers:

- Application Load Balancer**: Handles HTTP and HTTPS traffic. It routes requests from users to Lambda functions, API Gateways, or Amazon EC2 instances.
- Network Load Balancer**: Handles TCP, UDP, and TLS traffic. It routes requests from VPCs to ALBs, NLBs, or AWS Lambda functions.
- Gateway Load Balancer**: Handles traffic for third-party virtual appliances supporting GENEVE. It routes requests from users to GWLBs, which then route to various services.

Each section includes a 'Create' button. A red arrow points to the 'Create' button under the Application Load Balancer section.

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

The screenshot shows the 'Create Application Load Balancer' wizard. In the 'Basic configuration' step, the 'Load balancer name' field contains 'web-tier-external-lb' (with a red arrow pointing to it). The 'Scheme' section has 'Internet-facing' selected (radio button highlighted with a red arrow). Under 'IP address type', 'IPv4' is selected (radio button highlighted with a red arrow). A note at the bottom says 'A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.'

## Select the correct network configuration for VPC and **public** subnets

The screenshot shows the 'VPC' configuration step. In the 'Mappings' section, two Availability Zones are selected: 'us-west-1b' and 'us-west-1c'. For 'us-west-1b', the 'Subnet' dropdown is set to 'subnet-042290143e243f9a7' and 'Public-Subnet-AZ-1' (with a red arrow pointing to it). For 'us-west-1c', the 'Subnet' dropdown is set to 'subnet-00511a4ff95e0335d' and 'Public-Subnet-AZ2' (with a red arrow pointing to it). Both subnets are listed under 'Assigned by AWS'.

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 AWS Load Balancer configuration interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, search bar ('Search for services, features, blogs, docs, and more'), and region selector ('N. California'). Below the navigation is a 'Security groups' section with a sub-header 'Info' and a note: 'A security group is a set of firewall rules that control the traffic to your load balancer.' A dropdown menu titled 'Select security groups' contains one item: 'Internet-facing-lb-sg sg-072fe91349cba745f X'. A red arrow points to this item. Below this is a 'Create new security group' button. The main configuration area is titled 'Listeners and routing' with a sub-header 'Info' and a note: 'A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specification. You can specify multiple rules and multiple certificates per listener after the load balancer is created.' It shows a single listener configuration for 'HTTP:80':

- Protocol: HTTP
- Port: 80
- Default action: Forward to 'WebTierTargetGroup' (Target type: Instance, IPv4)

A red arrow points to the 'Forward to' dropdown, and another red arrow points to the 'Create target group' button.

## Launch Template

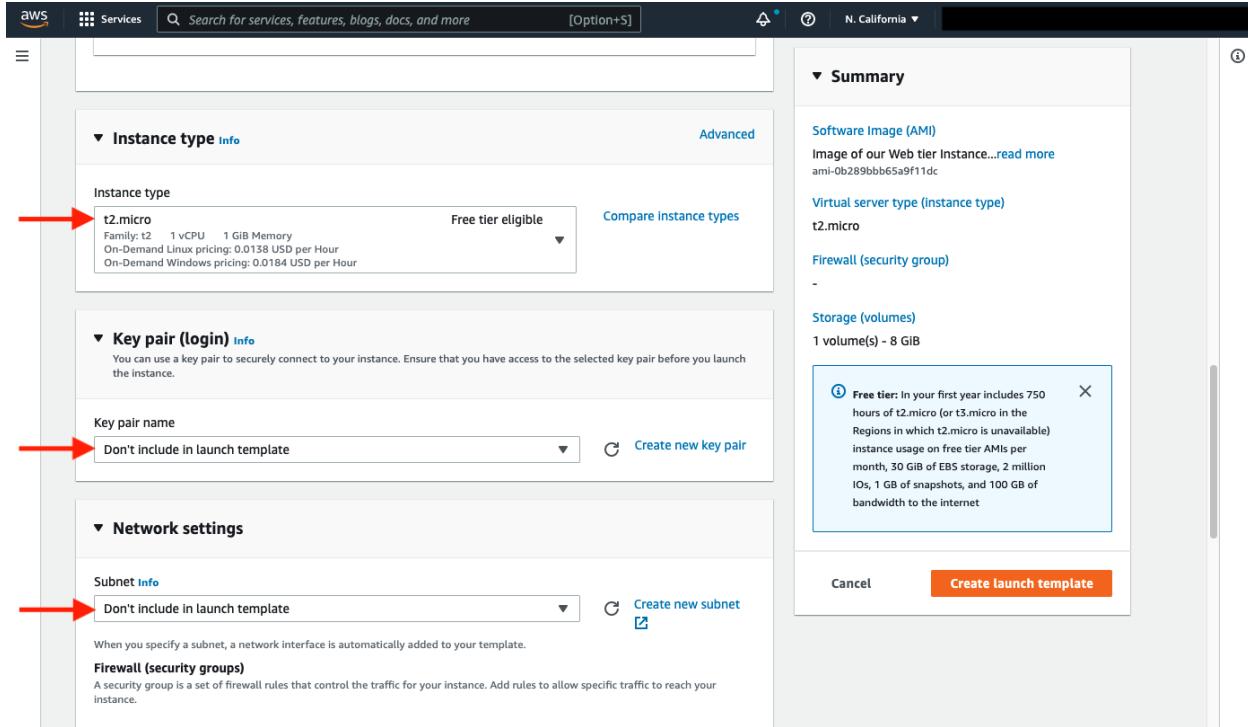
1. Before we configure Auto Scaling, we need to create a Launch template with the AMI we created earlier. On the left side of the EC2 dashboard navigate to **Launch Template** under **Instances** and click **Create Launch Template**

The screenshot shows the AWS EC2 Launch Templates page. On the left, there's a navigation sidebar with links like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with Launch Templates highlighted), Images, AMIs, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, and Network & Security. The main area shows a table with one entry: Launch template ID 'lt-0ef50c59ed0aa3255', Launch template name 'AppTierLaunchTemplate', Default version '1', and Latest version '1'. At the top right of the main area, there's a 'Create launch template' button.

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

This screenshot shows the 'Create launch template' wizard. In the 'Amazon machine Image (AMI)' section, the 'My AMIs' tab is selected. It shows three options: 'Don't include in launch template' (unchecked), 'Owned by me' (checked), and 'Shared with me'. Below this, a list of recent AMIs includes 'WebTierImage' (selected). To the right, there's a 'Summary' section with tabs for 'Software image (AMI)', 'Virtual server type (instance type)', 'Firewall (security group)', and 'Storage (volumes)'. A callout box provides details about the free tier for the selected AMI. At the bottom right of the wizard, there's a 'Create launch template' button.

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.



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

**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 groups Info**

Select security groups ▾

WebTiersSG sg-0d115f758786138a5 X  
VPC: vpc-0d2a264d3ba772982

Compare security group rules

Advanced network configuration

**Configure storage Info**

Advanced

1x 8 GiB gp2 Root volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage X

Add new volume

**Resource tags Info**

No resource tags are currently included in this template. Add a resource tag to include it in the launch template.

**Summary**

**Software Image (AMI)**  
Image of our Web tier Instance...read more  
ami-0b289bbb65a9f11dc

**Virtual server type (instance type)**  
t2.micro

**Firewall (security group)**  
WebTiersSG

**Storage (volumes)**  
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet

Create launch template

**Resource tags Info**

No resource tags are currently included in this template. Add a resource tag to include it in the launch template.

Add tag

50 remaining (Up to 50 tags maximum)

**Advanced details Info**

Purchasing option Info

Request Spot Instances  
Request Spot Instances at the Spot price, capped at the On-Demand price

IAM instance profile Info

aws-3tier-workshop-ec2-role  
arn:aws:iam::192462669998:instance-profile/aws-3tier-workshop-ec2-role

Create new IAM profile

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

Shutdown behavior Info

Don't include in launch template

Stop - Hibernate behavior Info

**Summary**

**Software Image (AMI)**  
Image of our Web tier Instance...read more  
ami-0b289bbb65a9f11dc

**Virtual server type (instance type)**  
t2.micro

**Firewall (security group)**  
WebTiersSG

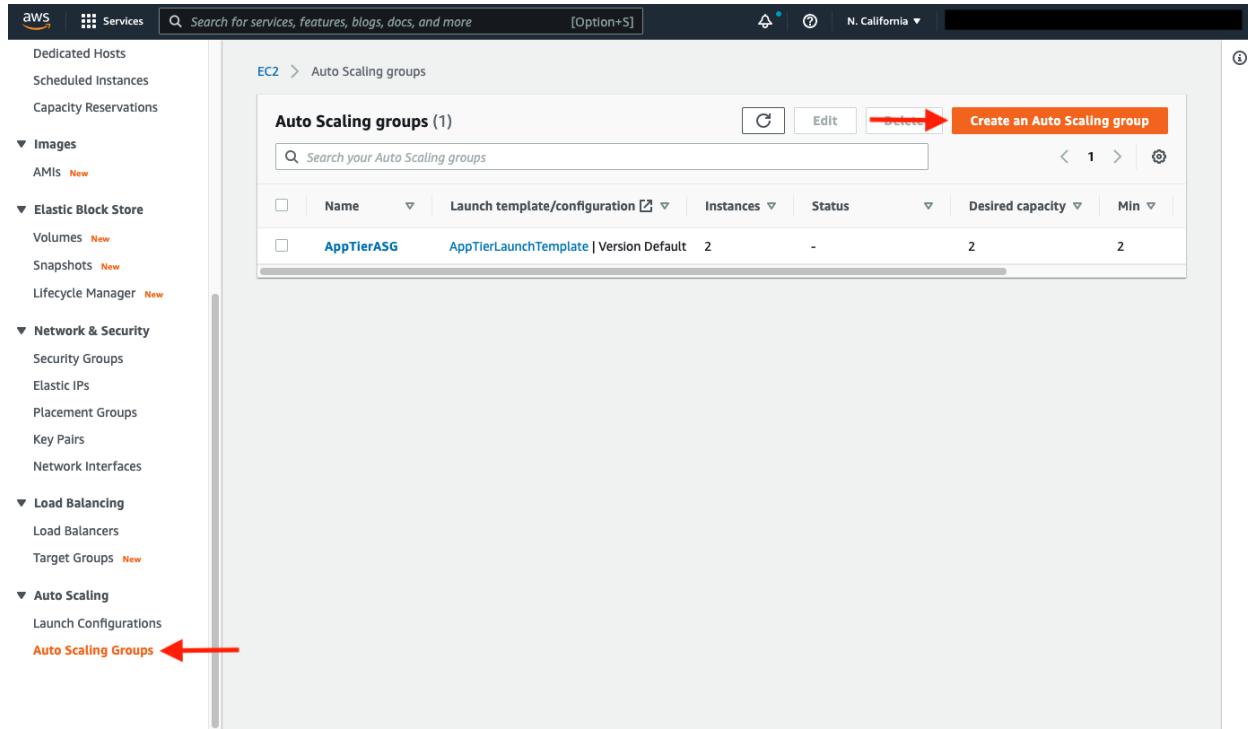
**Storage (volumes)**  
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet

Create launch template

## Auto Scaling

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



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

Search for services, features, blogs, docs, and more [Option+S] N. California

Dedicated Hosts  
Scheduled Instances  
Capacity Reservations  
**Images**  
AMIs New  
**Elastic Block Store**  
Volumes New  
Snapshots New  
Lifecycle Manager New  
**Network & Security**  
Security Groups  
Elastic IPs  
Placement Groups  
Key Pairs  
Network Interfaces  
**Load Balancing**  
Load Balancers  
Target Groups New  
**Auto Scaling**  
Launch Configurations  
**Auto Scaling Groups**

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1 Choose launch template or configuration

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

**Name**

Auto Scaling group name  
Enter a name to identify the group.  
**WebTierASG**

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

**Launch template** Info Switch to launch configuration

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

Create a launch template Version Default (1) Create a launch template version

Description - Launch template WebTierLaunchTemplate Instance type t2.micro

Step 2 Choose instance launch options

Step 3 (optional) Configure advanced options

Step 4 (optional) Configure group size and scaling policies

Step 5 (optional) Add notifications

Step 6 (optional) Add tags

Step 7 Review

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

Search for services, features, blogs, docs, and more [Option+S] N. California

Dedicated Hosts  
Scheduled Instances  
Capacity Reservations  
**Images**  
AMIs New  
**Elastic Block Store**  
Volumes New  
Snapshots New  
Lifecycle Manager New  
**Network & Security**  
Security Groups  
Elastic IPs  
Placement Groups  
Key Pairs  
Network Interfaces  
**Load Balancing**  
Load Balancers  
Target Groups New  
**Auto Scaling**  
Launch Configurations  
**Auto Scaling Groups**

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1 Choose launch template or configuration

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

**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-0d2a264d3ba772982 (awsthetierwor...)** 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

us-west-1b | subnet-04290143e243f9a7 (Public-Subnet-AZ-1) 10.0.0.0/24

us-west-1c | subnet-00511a4ff95e0335d (Public-Subnet-AZ2) 10.0.3.0/24

Create a subnet

Step 2 Choose instance launch options

Step 3 (optional) Configure advanced options

Step 4 (optional) Configure group size and scaling policies

Step 5 (optional) Add notifications

Step 6 (optional) Add tags

Step 7 Review

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

The screenshot shows the AWS Create Auto Scaling Group wizard at Step 6: Attach to an existing load balancer. On the left sidebar, under the 'Auto Scaling' section, 'Auto Scaling Groups' is highlighted in red. The main panel shows the following steps:

- Step 1:** Choose launch template or configuration
- Step 2:** Choose instance launch options
- Step 3 (optional):** Configure advanced options
- Step 4 (optional):** Configure group size and scaling policies
- Step 5 (optional):** Add notifications
- Step 6 (optional):** Add tags
- Step 7:** Review

In Step 6, there are two 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.

A red arrow points to the second option. Below it, the 'Attach to an existing load balancer' section shows:

Select the load balancers that you want to attach to your Auto Scaling group.

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

Choose from Classic Load Balancers

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

Select target groups ▾ C

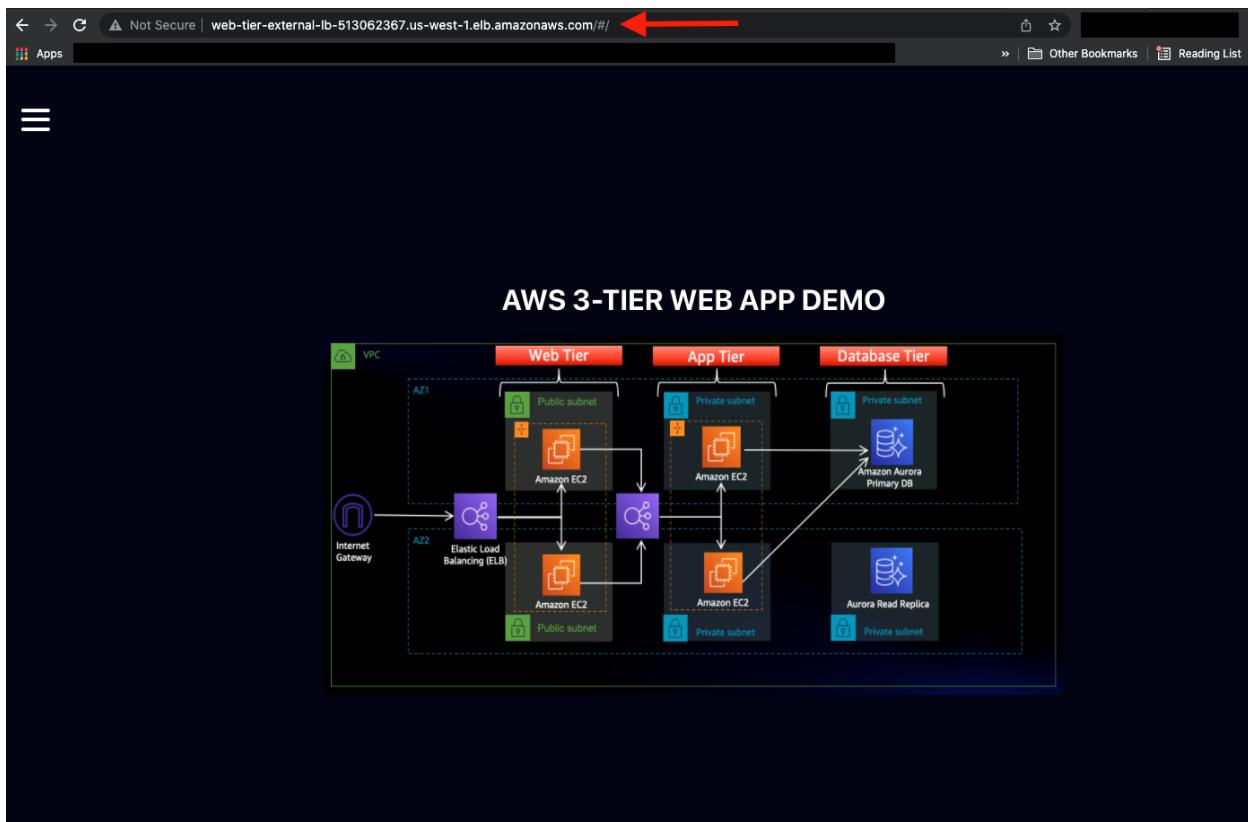
WebTierTargetGroup | HTTP Application Load Balancer: web-tier-external-lb

A red arrow points to the 'WebTierTargetGroup | HTTP Application Load Balancer: web-tier-external-lb' entry in the dropdown.

5. 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 AWS Management Console with the navigation bar at the top. On the left, there's a sidebar with various service links like Capacity Reservations, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. Under Auto Scaling, 'Auto Scaling Groups' is selected. The main content area is titled 'Configure group size and scaling policies'. It has a sub-section 'Step 4 (optional) Configure group size and scaling policies'. Below this, there are three input fields: 'Desired capacity' (set to 2), 'Minimum capacity' (set to 2), and 'Maximum capacity' (set to 2). Red arrows point to each of these three input fields. To the right of these fields is a section titled 'Scaling policies - optional' with two radio button options: 'Target tracking scaling policy' (unchecked) and 'None' (checked).

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.



*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!