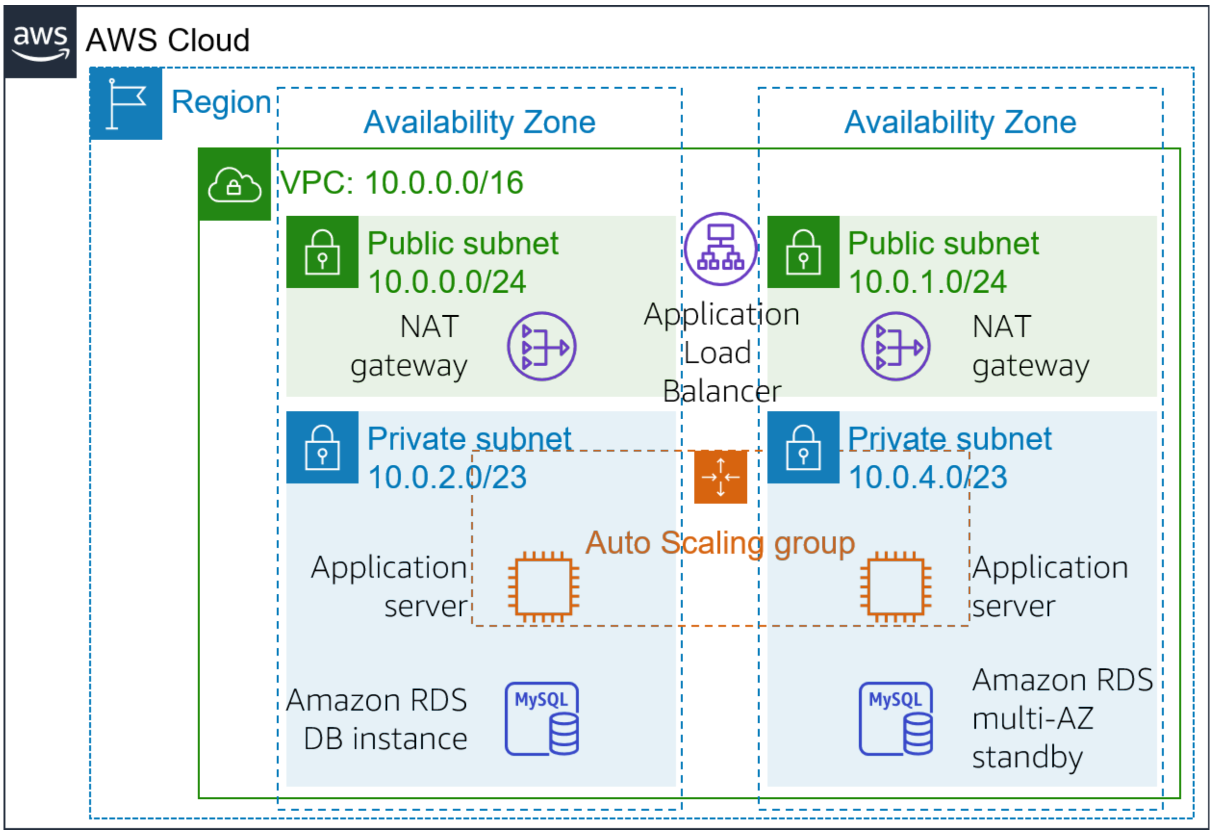
Highly Available Environment using Elastic Load Balancers & Autoscaling Groups

Type	AWS ARCHITECTURE
Tasks Achieved	- Creating and Application Load Balancer - Using an Autoscaling Group to allow scaling of EC2 Instances incase of a failure or in need of more resources

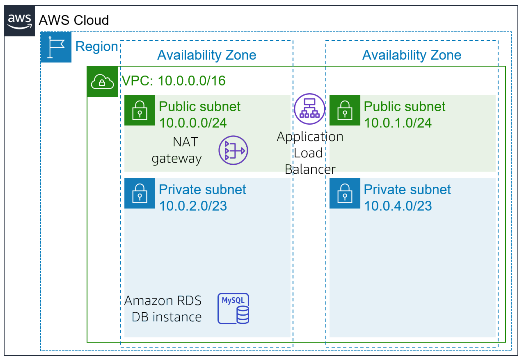
Tasks Achieved



▼ Creating Application Load Balancer - launched in the public subnets

▼ Creating a highly available DB

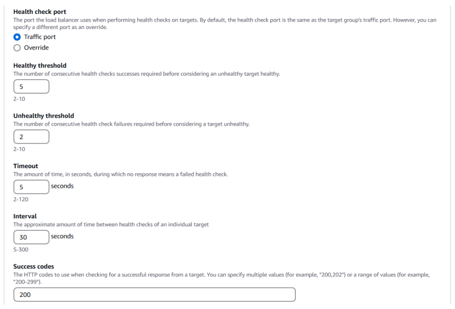
- Ensure you have a DB subnet group across two availability


AWS Cloud
Region
Availability Zone
Availability Zone
VPC: 10.0.0.0/16
Public subnet 10.0.0.0/24
NAT gateway
Application Load Balancer
Public subnet 10.0.1.0/24
Private subnet 10.0.2.0/23
Private subnet 10.0.4.0/23
Amazon RDS DB instance
MySQL

→ You will launch the ALB in the public subnets to as they are internet facing

→ The security group will only allow HTTP & HTTPS traffic

→ When setting the listeners you can create or choose from a target group. The target group will be instances which will have port 80 open from which the listeners from the Application Load Balancer will listen from. You can set the following in advance settings:



Health check port
The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.
[✓ Traffic port] Override

Healthy threshold
The number of consecutive health checks successes required before considering an unhealthy target healthy.
5
2-10

Unhealthy threshold
The number of consecutive health check failures required before considering a target unhealthy.
2
2-10

Timeout
The amount of time, in seconds, during which no response means a failed health check.
5 seconds
2-120

Interval
The approximate amount of time between health checks of an individual target
30 seconds
5-300

Success codes
The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").
200

→ When setting up the load balancer you choose the two availability zones to launch from as well as the subnet

You will require:

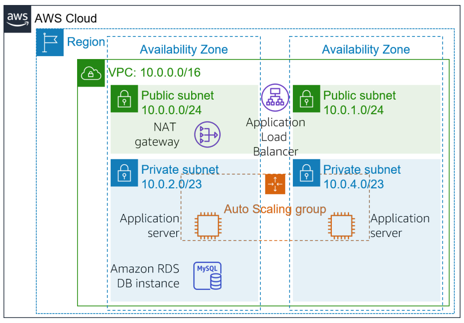
zones by creating a standby instance

- Edit already created DB to allow Multi-AZ deployment
- Remember as you make it highly available double the instances as well as the storage capacity

▼ Creating a highly available NAT Gateway in the case where the one in AZ 1 fails

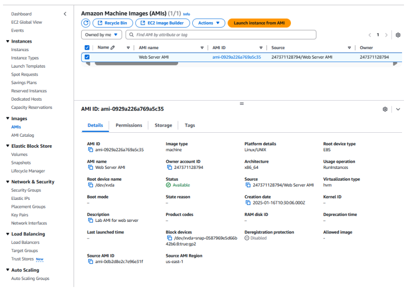
- Remember to allocate an elastic IP when creating
- Create a separate route table to associate the second private subnet to the created NAT gateway

- An autoscaling group that would automatically launch instances in the private subnets



1. Create an AMI from an existing instance

▼ Create an AMI



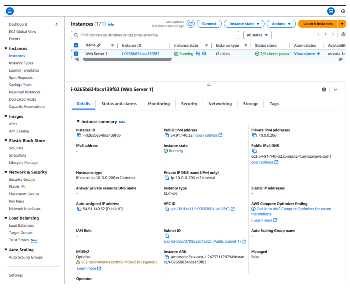
Amazon Machine Images (AMIs) (1/1)

Name	AMI name	AMI ID	Source	Owner
	Web Server AMI	ami-0929a226a769a5c35	247371128794/Web Server AMI	247371128794

AMI ID: ami-0929a226a769a5c35

AMI ID ami-0929a226a769a5c35	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name Web Server AMI	Owner account ID 247371128794	Architecture x86_64	Usage operation RunInstances
Root device name /dev/xvda	Status Available	Source 247371128794/Web Server AMI	Virtualization type hvm
Boot mode –	State reason –	Creation date 2025-01-16T10:30:06.000Z	Kernel ID –
Description Lab AMI for web server	Product codes –	RAM disk ID –	Deprecation time –
Last launched time –	Block devices /dev/xvda=snap-0587969e5d66b42b6:8:true:gp2	Deregistration protection Disabled	Allowed image -
Source AMI ID ami-0db2d8e2c7e96e31f	Source AMI Region us-east-1		

- An Image ID is a copy of another EC2 Instance. In this case Web Server 1



Instances (1/1) Last updated less than a minute ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Web Server 1	i-0265b834bca139f83	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a

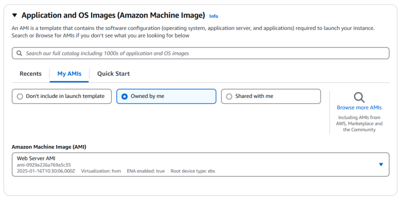
i-0265b834bca139f83 (Web Server 1)

▼ Instance summary

Instance ID i-0265b834bca139f83	Public IPv4 address 54.91.140.32 | open address	Private IPv4 addresses 10.0.0.206
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-54-91-140-32.compute-1.amazonaws.com | open address
Hostname type IP name: ip-10-0-0-206.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-0-206.ec2.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 54.91.140.32 [Public IP]	VPC ID vpc-091fae111c9066966 (Lab VPC)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-02a70709562c1e82c (Public Subnet 1)	Auto Scaling Group name –
IMDSv2 Optional EC2 recommends setting IMDSv2 to required | Learn more	Instance ARN arn:aws:ec2:us-east-1:247371128794:instance/i-0265b834bca139f83	Managed false
Operator		

2. Create a launch template

▼ Choose the AMI from which the instance will be created


Application and OS Images (Amazon Machine Image) Info

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

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

Recents | My AMIs | Quick Start

( ) Don't include in launch template [✓ Owned by me] ( ) Shared with me

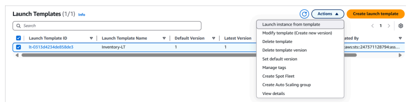
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)
Web Server AMI
ami-0929a226a769a5c35
2025-01-16T10:30:06.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

- You can add additional setting in the advanced settings section such as the IAM role and the user data

3. An autoscaling group - launched in the private subnets

▼ After creating the launch template you can create an autoscaling group


Launch Templates (1/1) Info

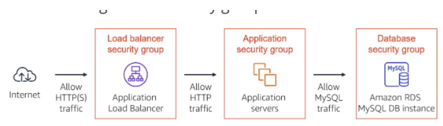
	Launch Template ID	Launch Template Name	Default Version	Latest Version	ated By
✓	lt-0313d4234de858de3	Inventory-LT	1	1	:aws:sts::247371128794:ass...

- **Choose instance launch options:** Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.
  - remember that the instances are being launched in

    the private subnets

- **Integrate with other services -** Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.
    - this is where you choose the already created application load balancer
- **Configure group size and scaling -** Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.
- You can choose to add notifications via creating a SNS topic

- Remember to edit the application server inbound rule to allow HTTP/HTTPS traffic from the load balancer as well as for the DB instance to allow DB traffic from the App server



- User data for an inventory website template

```
#!/bin/bash
# Install Apache Web Server and PHP
yum install -y httpd mysql
amazon-linux-extras install -y php7.2
# Download Lab files
wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/ILT-TF-200-ACACAD-20-EN/mod9-guided/scripts/inventory-app.zip
unzip inventory-app.zip -d /var/www/html/
# Download and install the AWS SDK for PHP
wget https://github.com/aws/aws-sdk-php/releases/download/3.62.3/aws.zip
unzip aws -d /var/www/html
# Turn on web server
```

```
chkconfig httpd on
service httpd start
```