

Autoscaling Group + Application Load balancer

App code

First, be sure to upload the app code on github to be able to clone it from the Launch Template when initializing the EC2 instances.

VPC and Subnets

Set up a VPC with 3 subnets, one public for the load balancer and two private for the autoscaling group and database (done in previous practices).

Launch template

Go to Launch Templates and use the following setup:

Step 2: Configure Template

AMI:

- Quick Start → Amazon Linux 2023 (latest)

Instance type:

- t2.micro

Key pair:

- Select jan26-key (from Week 1)

Network settings:

- Subnet: public-jan26 (for now - in production use private)
- Availability Zone: ap-south-1a

Security group:

- Create new: asg-app-sg
- Inbound rules:
 - SSH (22) from 0.0.0.0/0
 - Custom TCP (8000) from 0.0.0.0/0

Advanced details → User data:

Paste the user data script (modify GitHub URL to YOUR repo):

```
#!/bin/bash
sleep 30
pwd > /home/ec2-user/install-logs.txt
sudo yum install git -y
cd /home/ec2-user
git clone https://github.com/YOUR_USERNAME/jan26-bootcamp.git
cd jan26-bootcamp/week3/day1/app
chmod u+x run.sh
./run.sh
```

Launch Templates (1) Info							
Actions Create launch template							
	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By	Managed
<input type="checkbox"/>	lt-04bce0386ccfe3ad	jan26-week3-template	1	1	2026-02-16T05:06:49.000Z	arn:aws:iam::166337233599:root	false

Remember to enable public IP in order to test the Launch Template.

Now to be sure that everything is working can test launching the template following these steps:

Step 4: Test Launch Template

1. Go to Launch template → Actions → **Launch instance from template**
2. Use default settings
3. Launch

Verify:

```
ssh -i jan26-key.pem ec2-user@<PUBLIC_IP>

# Check logs
cat /home/ec2-user/install-logs.txt

# Check if app is running
curl http://localhost:8000
```



Test in browser: `http://<PUBLIC_IP>:8000`

If working, **terminate this test instance**.

Also is very useful

`cat /var/log/cloud-init-output.log`

to check init logs.

Autoscaling group

Lets create autoscaling group following next steps, first, ASG needs at least two subnets in different AZs:

Step 1: Create Second Public Subnet

ALB needs minimum 2 subnets in different AZs.

1. VPC → Subnets → **Create subnet**
2. Name: `public-jan26-2`
3. VPC: `jan26-vpc`
4. AZ: `ap-south-1b` (different from first subnet!)
5. CIDR: `10.0.3.0/24`

Associate with public route table:

1. VPC → Route Tables → Select `public-rt-jan26`
2. Subnet Associations → Edit
3. Add `public-jan26-2`
4. Save

You have successfully updated subnet associations for rtb-0aa1357f15e794022 / public-rt-jan26.

Route tables (1/4) Info

Last updated less than a minute ago Actions Create route table

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
public-rt-jan26	rtb-0aa1357f15e794022	2 subnets	-	No	vpc-04ff22a1f5bf0875 jan26-vpc	166537233399
-	rtb-0900d8f033fd51fb	-	-	Yes	vpc-0e8170e77d72b1e4	166537233399
private-rt-jan26	rtb-97c174859653e66d	subnet-02ab9f45b7e3dd...	-	No	vpc-04ff22a1f5bf0875 jan26-vpc	166537233399
-	rtb-0f58a134bf5b21dea	-	-	Yes	vpc-04ff22a1f5bf0875 jan26-vpc	166537233399

Now let's create the Auto Scaling Group following next setup:

Step 2: Create Auto Scaling Group

1. EC2 → Auto Scaling Groups → **Create**
2. Name: `jan26-week3-asg`
3. Launch template: `jan26-week3-template` (latest version)
4. Click **Next**

Network:

- VPC: `jan26-vpc`
- Subnets: Select `public-jan26` (the one matching your launch template AZ)

Load balancing:

- Select: **No load balancer** (we'll attach later)

Health checks:

- Keep defaults

Group size:

- Desired: `2`
- Minimum: `1`
- Maximum: `3`

Scaling policies:

- Target tracking scaling policy
- Metric: Average CPU utilization
- Target value: `70`

Finally after two - three minutes you should see two instances running.

Instances (2) Info

All states

Instance state = running

Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Mo
	i-0fa2fe42fe0ce62	Running	t3.micro	Initializing	View alarms +	us-east-1a	-	54.91.155.36	-	-	dis
	i-0d157d8131b6c4d6e	Running	t3.micro	Initializing	View alarms +	us-east-1a	-	54.167.16.45	-	-	dis

Target group

Create a target group following next steps:

🎯 PART 4: CREATE TARGET GROUP

Step 1: Create Target Group

1. EC2 → Target Groups → **Create**
2. Target type: **Instances**
3. Name: `jan26-week3-tg`
4. Protocol: HTTP
5. Port: 8000
6. VPC: `jan26-vpc`

Health check:

- Protocol: HTTP
- Path: `/`
- Port: 8000

Step 2: Register Targets

Don't manually select instances! (they're managed by ASG)

Just click **Next** then **Create target group**.

We'll connect ASG to this target group via ALB.

jan26-week3-tg						Actions ▾
Details						
Target type Instance	Protocol : Port HTTP: 8000	Protocol version HTTP1	VPC vpc-04f7f22a1fb0875			
IP address type IPv4	Load balancer None associated					
0 Total targets	0 Healthy	0 Unhealthy	0 Unused	0 Initial	0 Draining	
0 Anomalous						

Load Balancer

Now create the load balancer using the following steps:

Step 1: Create ALB

1. EC2 → Load Balancers → **Create**
2. Type: **Application Load Balancer**
3. Name: `jan26-week3-alb`

Scheme: Internet-facing

Network mapping:

- VPC: `jan26-vpc`
- Subnets: Select BOTH
 - `public-jan26` (ap-south-1a)
 - `public-jan26-2` (ap-south-1b)

Step 2: Create ALB Security Group

Create new security group:

- Name: `alb-sg`
- VPC: `jan26-vpc`
- Inbound rules:
 - HTTP (80) from 0.0.0.0/0
 - HTTPS (443) from 0.0.0.0/0

Select this security group for ALB.

Step 3: Configure Listener

Default action:

- Protocol: HTTP
- Port: 80
- Forward to: `jan26-week3-tg`

Step 4: Create ALB

Click **Create load balancer**

Wait 2-3 minutes for ALB to become **Active**.

jan26-week3-alb

Details

Load balancer type Application
Status Provisioning
VPC `vpc-04f7f22a1f5bf0875`
Scheme Internet-facing
Hosted zone `Z35SXDXTRQ7X7K`
Availability Zones `subnet-0eb45cfcb4da4af3b4f` us-east-1a (use1-az4)
`subnet-d796358d6cf71c66` us-east-1b (use1-az6)
Load balancer IP address type IPv4
Date created February 16, 2026, 07:02 (UTC+01:00)

Load balancer ARN `arn:aws:elasticloadbalancing:us-east-1:166337233599:loadbalancer/app/jan26-week3-alb/ed2119c06b52bb29`
DNS name info `jan26-week3-alb-254524075.us-east-1.elb.amazonaws.com (A Record)`

Remember to attach the ASG to the ALB.

Edit jan-week3-asg

Load balancing - optional
Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

Load balancers

Application, Network or Gateway 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 `jan26-week3-tg | HTTP`
Application Load Balancer: `jan26-week3-alb`

Classic Load Balancers

Create and attach new load balancers

Add a new load balancer

Cancel Update

In the target group should see the registered targets:

The screenshot shows the AWS Elastic Load Balancing console for a target group named 'jan26-week3-tg'. The top section displays summary statistics: 1 Total targets, 1 Healthy (green), 0 Unhealthy (red), 0 Unused, 0 Initial, and 0 Draining. Below this is a table titled 'Distribution of targets by Availability Zone (AZ)' with one row for 'us-east-1a' showing 1 healthy target. The main table lists one registered target: 'i-0d157d8131b6c4d6e' with port 8000, zone 'us-east-1a', health status 'Healthy' (green), and administrative override 'No override'.

DNS

Configure HTTP connections (DNS) using the following steps:

PART 6: CONFIGURE DNS

Step 1: Create Subdomain Record

1. Route 53 → Hosted zones → Your domain
2. **Create record**
3. Record name: `week3` (creates `week3.yourdomain.com`)
4. Record type: **A**
5. **Alias:** Yes
6. Route traffic to:
 - o Application and Classic Load Balancer
 - o Region: `ap-south-1`
 - o Select your ALB: `jan26-week3-alb`
7. Create

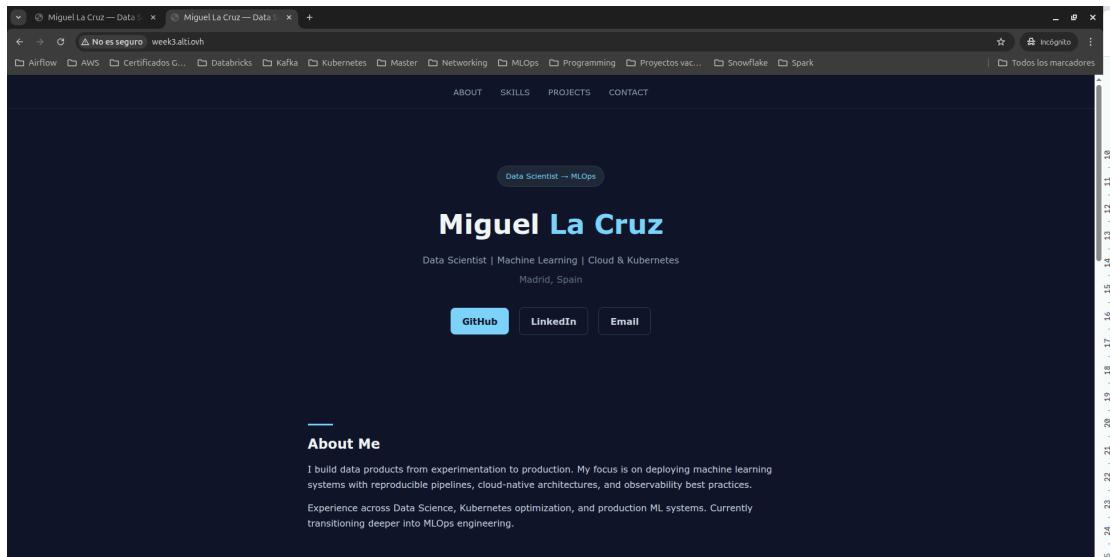
Step 2: Test HTTP

Wait 2-3 minutes for DNS propagation.

Visit: <http://week3.yourdomain.com>

Should see your Flask app (but "Not Secure").

Now should be able to connect to the http domain:



HTTPS

Now to add HTTPS:

🔒 PART 7: ADD HTTPS

Step 1: Request SSL Certificate

1. Certificate Manager → **Request certificate**
2. Domain: `week3.yourdomain.com`
3. Validation: DNS validation
4. **Do NOT enable export** (keeps it free!)
5. Request

Step 2: Validate Certificate

1. Click certificate
2. **Create records in Route 53**
3. Wait for status: **Issued** (~5-10 minutes)

Step 3: Add HTTPS Listener to ALB

1. Load Balancers → Select your ALB
2. **Listeners tab** → **Add listener**
3. Protocol: **HTTPS**
4. Port: `443`
5. Default action: Forward to `jan26-week3-tg`

Secure listener settings:

- Security policy: Recommended
 - Default SSL certificate: **From ACM**
 - Select: `week3.yourdomain.com`
6. **Add**

Overall architecture

