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`

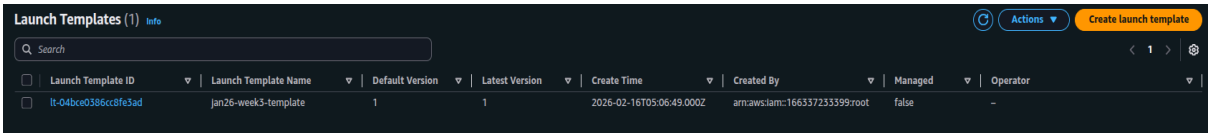
Security group:

- 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
```



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:

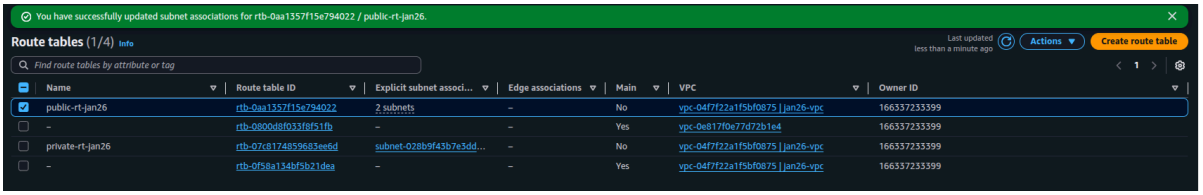
Step 1: Create Second Public Subnet

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



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

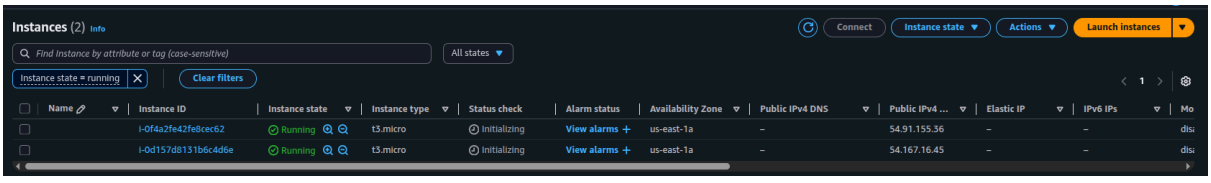
Group size:

- 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.



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:

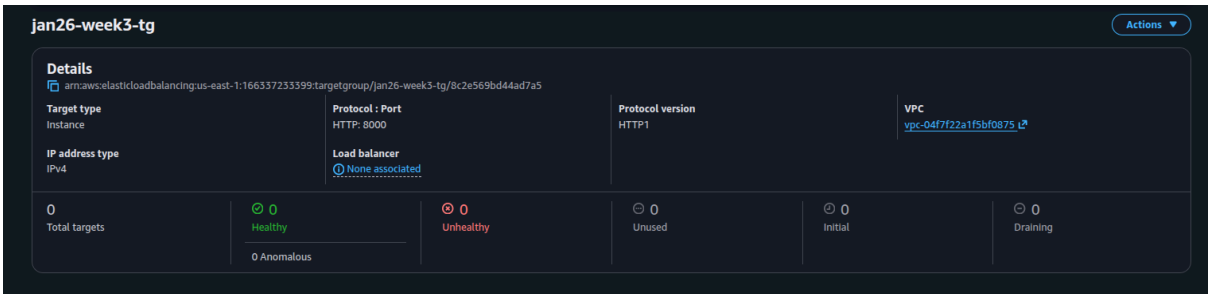
- Protocol: HTTP
- Path: `/`
- Port: `8000`

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.



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

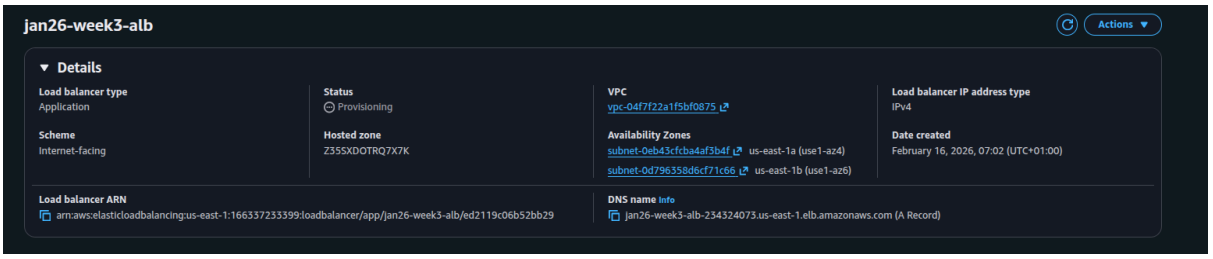
Default action:

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

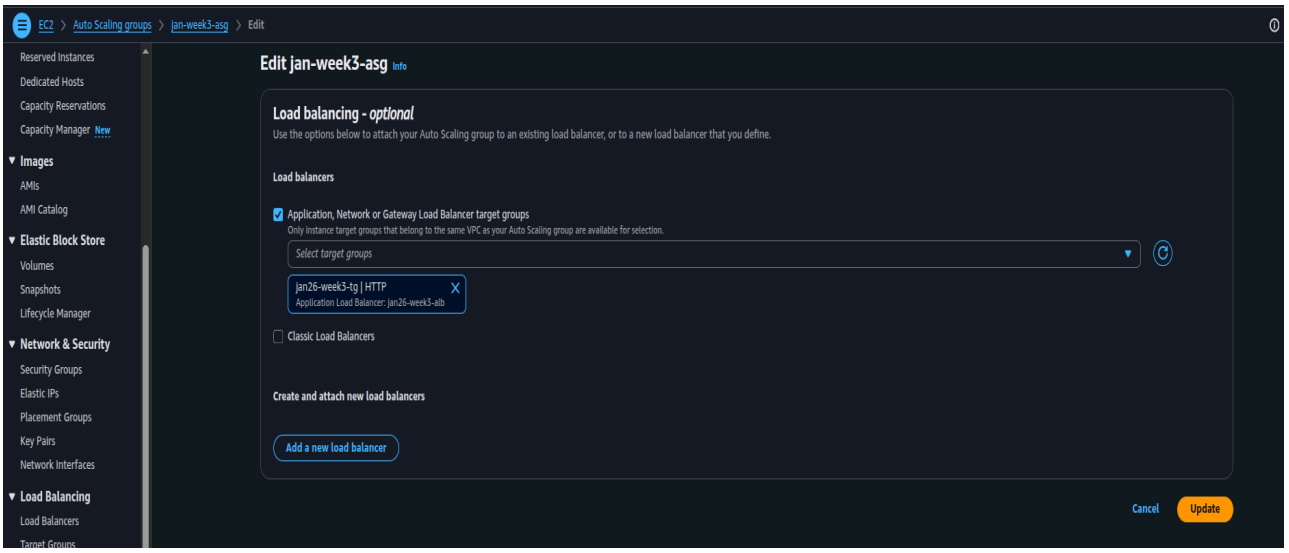
Step 4: Create ALB

Click **Create load balancer**

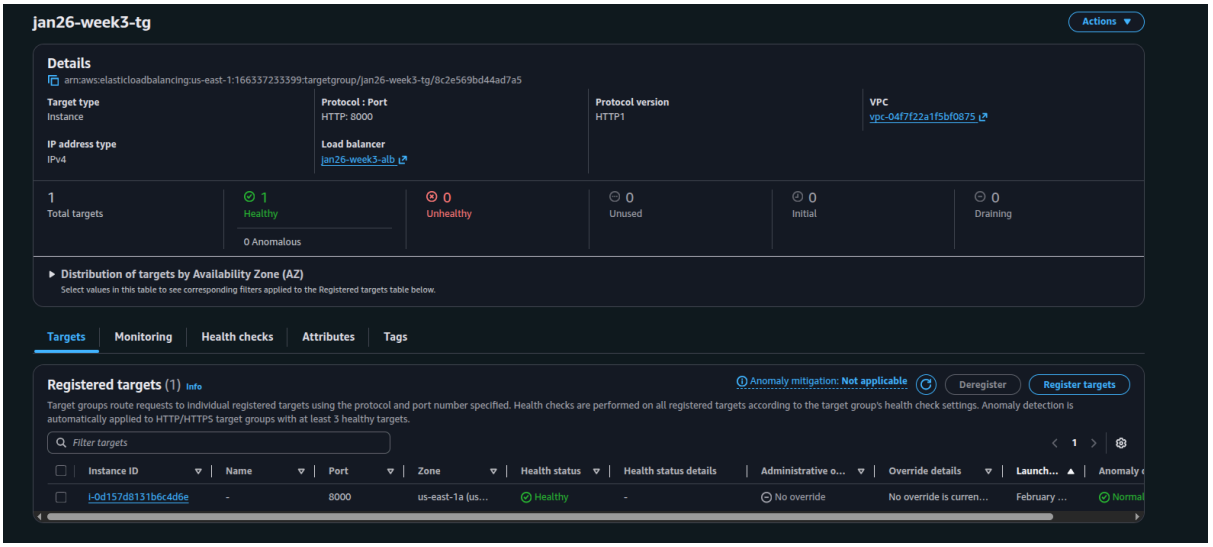
Wait 2-3 minutes for ALB to become **Active**.



Remember to attach the ASG to the ALB.



In the target group should see the registered targets:



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:
   - Application and Classic Load Balancer
   - Region: `ap-south-1`
   - Select your ALB: `jan26-week3-alb`
7. Create

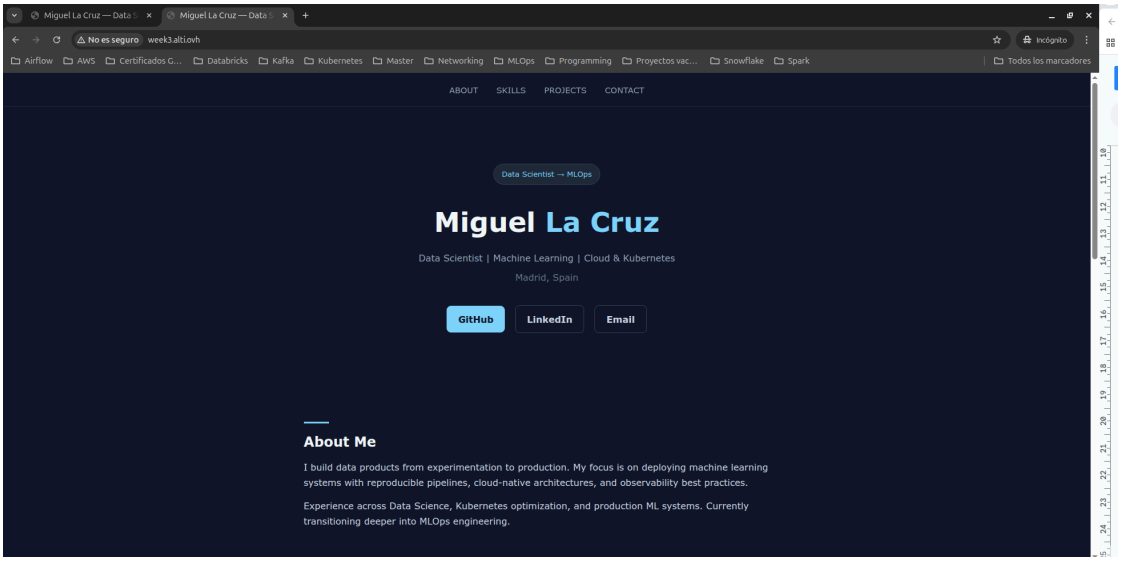
Step 2: Test HTTP

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