Case - 1 →

https://github.com/mactesting/case-1

Step - 1 →

- Replace the keyname, one can ssh into vm with public ip, ec2-user
- Connect with lb on browser you can see

What the Terraform builds (mapped to your checklist)

1. **VPC & Networking**
   - **VPC** (e.g., `10.0.0.0/16`)
   - **2 public subnets** (for internet-facing instances or a load balancer)
   - **Internet Gateway + public route table** (`0.0.0.0/0 → igw`)
   - (Optional but common: 2 private subnets if you put instances behind an ALB)
2. **Security Groups**
   - **web_sg**: inbound **80/443** from world (or your IPs), **22/SSH** from your office IP; all egress allowed.
     (Optional) **alb_sg**: open 80/443 from world; the instance SG then allows inbound **from alb_sg** only.
3. **EC2 + EBS (application + persistent data)**

- **Launch Template** (or Launch Configuration) with:
  - AMI + instance type (e.g., Amazon Linux 2023, `t3.micro`/`t3.small`)
    User-data that installs a simple web server and mounts an **EBS data volume**
  - **Block device mappings** to create/attach an **additional EBS volume** (e.g., 10–20 GiB, gp3) for app data at `/mnt/data`.
- The user-data usually does:
  - `mkfs` and mount the data volume (first boot)
  - write an `index.html` that includes the **instance-id** so you can see which instance served the request

4. **Auto Scaling**
   - **Auto Scaling Group (ASG)** spanning two subnets, min/desired/max (e.g., 1/1/4).
   - **Scaling policy** via CloudWatch:
     - **Target Tracking (CPU)**: keep avg CPU at, say, **40%**, or
     - **Step scaling** on **CPUUtilization** alarms (e.g., >60% scale out, <20% scale in).
5. **(Recommended) Load Balancer + Target Group**
   - While your list didn't mention it explicitly, a scalable web app practically needs an **ALB** so traffic can be distributed to the changing set of instances in the ASG.
   - **ALB** in public subnets → **HTTP listener :80** → **Target Group** (instance or IP targets, health checks on `/`).
   - The ASG registers instances to the target group.
6. **Route 53**
   - A **public hosted zone** for your domain (or use an existing one).
   - An **A/AAAA alias record** mapping **`app.yourdomain.com`** → **ALB DNS** (recommended).
     - If you truly skip ALB, you'd need a different pattern (e.g., Route 53 → NLB or a single fixed instance); aliasing to

individual, ephemeral instance IPs won't work with autoscaling.

7. **CloudWatch**

    - **Metrics & Alarms** on **CPUUtilization** for the scaling policy.
    - (Optional) **Log group** for your app if you send logs via `cloudwatch-agent` or systemd-journald export.

How the data-persistence works (EBS in an ASG)

- The Launch Template's **block_device_mappings** creates **one EBS data volume per instance** and mounts it at boot.
- **EBS is per-instance**; it **does not** auto-move between ASG instances. It persists for the life of that instance (and can persist after termination only if you set `delete_on_termination = false`, then reattach manually/with automation in the *same AZ*).
- For **shared or truly persistent** state across scaling events, consider **EFS** or an external DB/S3. For this scenario, EBS is fine for demonstrating "persistent across reboots / stop-start" and "per-instance persistence".

Test Plan (hands-on demo friendly)

0) Pre-flight (2–3 min)

Goal: Make sure the app is up and you can log in to instances.

1. **Find the Load Balancer URL**

- Console → **EC2** → left nav **Load Balancers** → select your ALB → **Description** → **DNS name**.
- Open it in a browser. You should see:
  Hello from Auto Scaling EC2! and **Served by: <hostname>** (this line changes between requests when multiple instances are in service).

2. **Connect to an instance**

- Console → **EC2** → **Auto Scaling Groups** → pick your group → **Activity** tab → click an **Instance ID** → **Connect**.
- Prefer **Session Manager** (works because your instances have the SSM role). Click **Connect** → you get a shell.
- (Alt: SSH) From your machine:
  `ssh -i <your-key.pem> ec2-user@<EC2 Public IP or DNS>`

1) MyWebApp (Nginx) — "hello page" (2–3 min)

What to show: App works; the page includes which EC2 served it.

Console path

- EC2 → **Load Balancers** → select ALB → **Listeners** → **View/edit rules** on port 80 (show default forward to target group).
- EC2 → **Target Groups** → select the group → **Targets** tab → show instances **healthy**.

Browser

- Refresh the ALB URL several times → watch **"Served by: …"** change → proves load balancing across instances.

Troubleshooting commands (on an instance via SSM/SSH)

```
# NGINX status
```

```
sudo systemctl status nginx

# See the bootstrap log written by user_data.sh

sudo tail -n 50 /var/log/bootstrap.log

# Page content

sudo cat /usr/share/nginx/html/index.html

hostname -f   # this is what your page prints after
"Served by:"
```

2) Auto Scaling (manual + policy) (6–8 min)

What to show: ASG grows/shrinks; LB starts sending traffic to the new instance(s).

A) Manual scale-out

- **Console → EC2 → Auto Scaling Groups → select group → Edit (or Automatic scaling → Desired capacity overrides).**
- **Set Desired capacity from e.g. 2 → 3 → Update.**
- **Go to Activity tab → watch instance launch events.**
- **EC2 → Instances → filter by your ASG → see the new instance reach 2/2 checks passed and target becomes healthy in Target Group.**
- **Browser: refresh your ALB URL until Served by shows the new hostname.**

B) Policy-based scale-out (CPU)

If you already have a scaling policy hooked to a CloudWatch alarm, you can trigger load:

Generate CPU load on one instance (choose any one)

```
# Try stress-ng (AL2023)

sudo dnf -y install stress-ng || true

sudo stress-ng --cpu 2 --timeout 180 &



# If stress-ng not available, fallback:

yes > /dev/null &   # start one or two of these in background

yes > /dev/null &
```

- **Console → CloudWatch → Alarms → open the CPU alarm → watch it go ALARM.**
- **Console → EC2 → Auto Scaling Groups → Activity → see a new instance launched by policy.**
- **Browser: refresh your ALB URL; show additional Served by hostnames.**

(When done, kill the load: `pkill yes` or wait for stress-ng timeout.)

3) Data Persistency with EBS (per-instance) (5–6 min)

What to explain: Your user-data mounts the extra EBS disk at **/data** (XFS). Data on **this instance** persists across reboots/stop-start because it's on EBS. (If an ASG **terminates** an instance and creates a new one, that's a new disk unless you've configured re-attach. For cross-instance/shared persistence you'd typically use **EFS/RDS**. For the demo, we'll show **persistence across reboot**.)

On an instance (SSM/SSH):

```
# Verify the mount
lsblk
df -h | grep /data
cat /etc/fstab | grep /data

# Write data
echo "Persistent data test $(date)" | sudo tee /data/test.txt
sync

# Reboot the instance (safe under ASG)
sudo reboot
```

After the instance comes back (reconnect):

```
# Confirm the file is still there
cat /data/test.txt
```

Talk track: “Because this is an EBS volume, data persisted across the reboot. If the ASG replaces this instance, a fresh EBS is attached to the new one unless we implement a reattach/lifecycle hook. For shared app data across instances, we’d use EFS/S3/DB.”

- (Trainer tip) Show that **root** disk content resets if you replace the instance, but **EBS data** sticks across reboots.

  Caveat: If the ASG replaces the instance, that *instance’s* volume won’t follow automatically. Call this out to the audience and recommend EFS for shared persistence.

4) Quicker rollout of a new webapp version (rolling update) (7–9 min)

Goal: Change the Nginx homepage via Launch Template user data and do an Instance Refresh for a rolling replacement with zero downtime.

1. **Prepare a small change (what to say):**
    - **“We’ll change the `<h1>` to ‘New Version v2’ in user data so every new instance boots with the updated page.”**
2. **Update Launch Template**
    - **Console → EC2 → Launch Templates → open your LT → Actions → Modify template (Create new version).**

Scroll to Advanced details → User data.
Find the line that writes the HTML; change:

```
<h1>Hello from Auto Scaling EC2!</h1>
```

to

```
<h1>Hello from Auto Scaling EC2! (v2)</h1>
```

    - **Create template version.**
    - **Back in the ASG: EC2 → Auto Scaling Groups → your ASG → Details → Edit → set Launch template version to the new version → Update.**
3. **Start a rolling Instance Refresh**
    - **In the ASG page → Instance refresh tab → Start instance refresh.**
    - **Strategy: Keep defaults (Min healthy percentage 90–100, Warmup 300s if health checks need time) → Start.**
    - **Watch instances replace one by one.**
4. **Browser verification**
    - **Keep refreshing the ALB URL while refresh progresses. Old pages will gradually be replaced; once a new instance is in service you’ll see (v2).**

(Tip: Also show Target Group → Targets to highlight deregistration/registration and health check stabilization.)

5) Monitoring & Alarms (4–6 min)

Console tour

- **CloudWatch → Metrics → EC2/Per-Instance Metrics: open CPUUtilization for your instances (or AutoScaling metrics).**
- **CloudWatch → Alarms: open your scaling alarm; point out OK/ALARM history and Actions linked to the ASG policy.**
- **Auto Scaling Group → Activity: correlate activity entries with alarm state changes.**

Quick log check for troubleshooting

```
# user-data bootstrap log (very useful)

sudo tail -n 100 /var/log/bootstrap.log

# NGINX access/error logs (AL2023)

sudo tail -n 50 /var/log/nginx/access.log

sudo tail -n 50 /var/log/nginx/error.log
```

6) Route 53 DNS mapping (3–5 min)

Goal: Friendly name (e.g., `mywebapp.yourdomain.com`) → ALB.

- **Console → Route 53 → Hosted zones → open your zone.**
- **Create record:**

    - **Record name: `mywebapp`**
    - **Record type: A – IPv4 address**
    - **Alias: Yes**
    - **Alias target: choose your ALB (appears in dropdown)**
    - **Routing policy: Simple**
    - **TTL: 60 seconds (demo-friendly)**
    - **Create records**
- **Test: browse to `http://mywebapp.yourdomain.com`.**
  (If it doesn't resolve immediately, wait 1–2 minutes or lower TTL ahead of time.)

7) Security (Security Groups) (3–5 min)

Show the rules and prove they work.

- Console → **EC2** → **Security Groups** → open the SG used by your instances (or ALB).
- **Inbound rules** (typical demo setup):
    - **HTTP (80)** from **0.0.0.0/0** (public web traffic)
    - **SSH (22) restricted to your IP** (click **My IP** to auto-fill)
- Save changes.
- **Prove it:**
    - From your machine, SSH works.
    - Ask a colleague (or your mobile hotspot on a different IP) to try SSH → **denied**.
    - App remains reachable on **HTTP 80** via ALB.

(Optional) WAF note: If you attached WAF to the ALB, briefly show a blocking rule in **WAF** → **Web ACLs**.

Handy one-liners you can paste during the demo

Check which instance served your last request

```
curl -s http://<ALB_DNS_or_Route53_Name>/ | grep "Served by"
```

Confirm /data is EBS-backed and mounted

```
lsblk

df -h | grep /data

sudo blkid | grep -v nvme0n1   # shows the extra disk UUID
```

Simulate CPU load (to trigger scaling)

```
sudo dnf -y install stress-ng || true

sudo stress-ng --cpu 2 --timeout 240 &

# or

yes > /dev/null &  # run 1–2 of these; kill with: pkill yes
```

Quickly change homepage on ONE instance (for a “hotfix” demo)

```
sudo sed -i 's/Hello from Auto Scaling EC2!/Hello from Auto Scaling EC2! (hotfix)/' /usr/share/nginx/html/index.html

sudo systemctl reload nginx
```

(Then explain why this isn't durable and why we use Launch Template + Instance Refresh for proper rollouts.)

Troubleshooting checklist (use when something doesn't show up)

- **ALB shows 5xx:**
    - **Target Group → Targets must be healthy; Health checks path `/` on port 80.**
    - **On instance: `sudo systemctl status nginx`; `sudo ss -ltnp | grep :80`.**
- **User data didn't run:**
    - **Check `sudo cat /var/log/bootstrap.log`.**
    - **Ensure Launch Template has the updated User data *and* the ASG is using that LT version.**
- **Scale policy didn't fire:**
    - **CloudsWatch alarm threshold too high? Reduce it temporarily for demo, or extend evaluation periods.**
    - **Ensure alarm Actions target your ASG policy.**
- **/data not mounted:**
    - **`lsblk` should show a second disk (non-root).**
    - **`cat /etc/fstab` should have a line for `/data`.**

- Try `sudo mount -a`.
- If no extra disk exists, adjust the Launch Template block device mappings to add a non-root EBS volume.

Optional 2-minute closing

- Scale-in: set Desired back to original (e.g., 3 → 2). Show graceful deregistration in Target Group.
- Health-check resiliency: Stop NGINX on one instance (`sudo systemctl stop nginx`) → watch Target go unhealthy and traffic still flows via healthy instances.