

# STEP-BY-STEP IMPLEMENTATION

## Step 1: Setting Up the VPC — shloka-canary-task-vpc (Oregon, 10.0.0.0/16)

### What was done:

A dedicated VPC was created with four subnets (two public, two private), routing tables, an Internet Gateway, and a NAT Gateway to support a highly available canary deployment architecture.

### Why was it done:

Canary deployments require reliable, isolated network environments where private EC2 instances run behind an ALB.

- **Public subnets** → ALB + Bastion
- **Private subnets** → Application servers (v1 and v2)

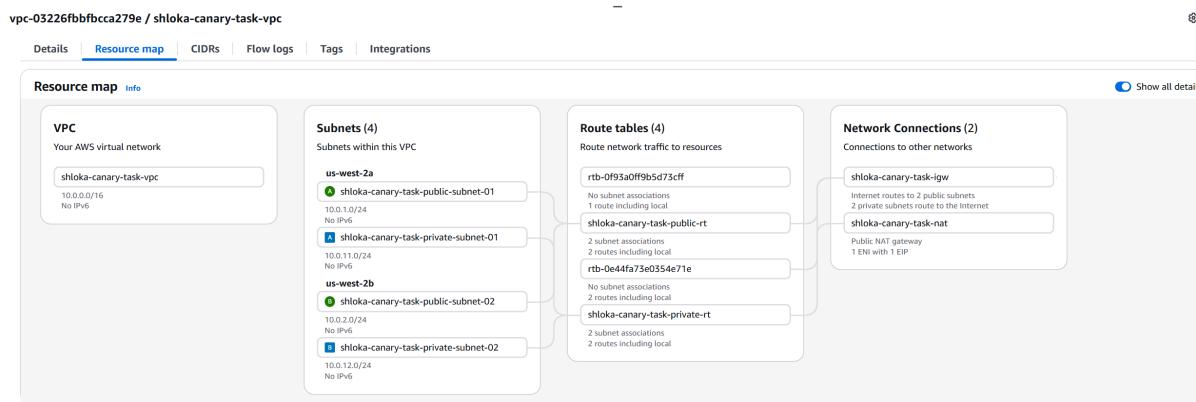
NAT + IGW are needed for controlled internet access while keeping servers private.

### How was it done:

- Created a VPC: `shloka-canary-task-vpc` with CIDR **10.0.0.0/16**
- Created four subnets:

| Subnet Name                          | AZ         | CIDR         | Type    |
|--------------------------------------|------------|--------------|---------|
| shloka-canary-task-public-subnet-01  | us-west-2a | 10.0.1.0/24  | Public  |
| shloka-canary-task-private-subnet-01 | us-west-2a | 10.0.11.0/24 | Private |
| shloka-canary-task-public-subnet-02  | us-west-2b | 10.0.2.0/24  | Public  |
| shloka-canary-task-private-subnet-02 | us-west-2b | 10.0.12.0/24 | Private |

- Created route tables:
  - `shloka-canary-task-public-rt` → route to **IGW**
  - `shloka-canary-task-private-rt` → route to **NAT Gateway**
- Attached:
  - Internet Gateway → `shloka-canary-task-igw`
  - NAT Gateway → `shloka-canary-task-nat`
- Associated all subnets to correct RTs.



## Step 2 - Launching EC2 Instances (Version 1—Stable App Servers)

### What was done:

Launched three EC2 instances:

- One **Bastion host** in public subnet
- Two **application servers (v1)** in private subnets

Nginx was installed and configured to serve Version 1 of the app.

### Why was it done:

- Bastion → secure SSH access into private instances
- v1 instances → represent the *current production* version
- These servers form the “old TG” for canary comparison

## How was it done:

Instances launched:

- shloka-canary-task-bastion → public-subnet-01
- shloka-canary-task-server01 → private-subnet-01
- shloka-canary-task-server02 → private-subnet-02

User data used to configure Nginx (Version 1 HTML):

```
#!/bin/bash
# version 1

yum -y update
yum -y install nginx

systemctl enable nginx
systemctl start nginx

cat << 'EOF' > /usr/share/nginx/html/index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Shloka Canary Page v1</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            background: linear-gradient(135deg, #0f2027, #203a43, #2c5364);
            font-family: "Inter", Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            color: #fff;
        }
    </style>
</head>
<body>
</body>
</html>
```

```
.card {  
    background: rgba(255,255,255,0.1);  
    padding: 50px 70px;  
    border-radius: 18px;  
    backdrop-filter: blur(8px);  
    box-shadow: 0 10px 35px rgba(0,0,0,0.3);  
    text-align: center;  
}  
  
h1 {  
    font-size: 3rem;  
    margin: 0 0 10px 0;  
    font-weight: 700;  
    letter-spacing: 1px;  
}  
  
p {  
    margin: 0;  
    font-size: 1.25rem;  
    opacity: 0.9;  
}  
  
.tag {  
    margin-top: 20px;  
    display: inline-block;  
    padding: 6px 14px;  
    background: rgba(255,255,255,0.2);  
    border-radius: 12px;  
    font-size: 0.9rem;  
    letter-spacing: 0.5px;  
}  
  
</style>  
</head>  
<body>  
    <div class="card">  
        <h1>Canary Deployment</h1>  
        <p>Instance is live and serving.</p>  
        <div class="tag">Version 1</div>  
    </div>
```

```
</body>
</html>
EOF

systemctl restart nginx
```

## Step 3 - Creating Target Group for Version 1

### What was done:

Created `shloka-canary-task-v1-TG` and registered both private server instances.

### Why was it done:

This target group will serve **90% traffic** during canary deployment and acts as the stable production environment.

### How was it done:

- Created Target Group → name: `shloka-canary-task-v1-TG`
- Protocol: HTTP 80
- Health check path: `/`
- Registered:
  - `server01`
  - `server02`

## Step 4 - Creating the Application Load Balancer

### What was done:

Created ALB with listeners, attached public subnets, configured SG, and forwarded traffic to TG-v1.

### Why was it done:

The ALB distributes traffic across v1 and (later) v2 instances and enables weighted canary deployment.

### How was it done:

- ALB name → `shloka-canary-task-alb`
- Subnets → both public subnets
- Security group → allow inbound **HTTP:80**
- Default action → forward to `shloka-canary-task-v1-TG`
- Updated private instance SG → allow inbound **80 from ALB SG**

Target group: shloka-canary-task-v1-TG

[Details](#) [Targets](#) [Monitoring](#) [Health checks](#) [Attributes](#) [Tags](#)

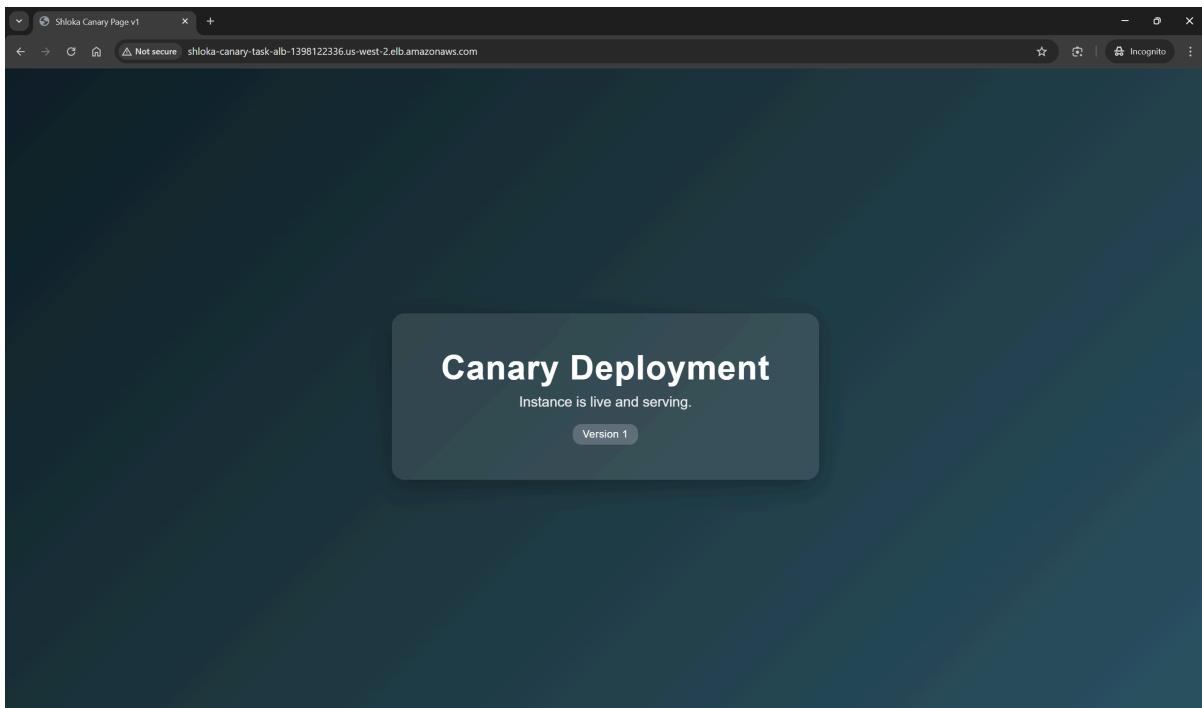
**Details**

arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63

|                         |   |                           |   |               |
|-------------------------|---|---------------------------|---|---------------|
| Target type<br>Instance | Protocol : Port<br>HTTP: 80                             | Protocol version<br>HTTP1 | VPC<br><a href="#">vpc-03226fbfbcca279e</a> |               |
| IP address type<br>IPv4 | Load balancer<br><a href="#">shloka-canary-task-alb</a> |                           |   |               |
| 2<br>Total targets      | 2<br>Healthy  | 0<br>Unhealthy            | 0<br>Unused                                 | 0<br>Initial  |
| 0 Anomalous             |   |                           |   | 0<br>Draining |

► Distribution of targets by Availability Zone (AZ)  
Select values in this table to see corresponding filters applied to the Registered targets table below.

Healthy Instances - SUCCESS



Web page loading on ALB DNS - SUCCESS

## Step 5 - Launching Version 2 Instance (Canary Instance)

### What was done:

Launched a separate EC2 instance configured to serve Version 2 of the application.

### Why was it done:

This instance represents the **canary version**, receiving only 10% of ALB traffic to safely test changes

### How was it done:

Instance name → `shloka-canary-task-server-test`

- Launched in private-subnet-01
- User data for Nginx serving **Version 2** HTML:

```
#!/bin/bash
# version 2

yum -y update
yum -y install nginx

systemctl enable nginx
systemctl start nginx

cat << 'EOF' > /usr/share/nginx/html/index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Shloka Canary Page v2</title>
    <style>
        body {
            margin: 0;
            height: 100vh;
```

```
        background: radial-gradient(circle at center, #ffe29f, #ffa99f, #f719a);
        font-family: "Inter", Arial, sans-serif;
        display: flex;
        justify-content: center;
        align-items: center;
        color: #222;
    }
.panel {
    background: rgba(255,255,255,0.7);
    padding: 45px 65px;
    border-radius: 20px;
    backdrop-filter: blur(10px);
    box-shadow: 0 8px 25px rgba(0,0,0,0.2);
    text-align: center;
}
h1 {
    margin: 0 0 12px;
    font-size: 2.8rem;
    font-weight: 700;
}
p {
    margin: 0;
    font-size: 1.2rem;
    opacity: 0.85;
}
.version {
    margin-top: 25px;
    display: inline-block;
    padding: 8px 18px;
    background: #222;
    color: #fff;
    border-radius: 14px;
    font-size: 0.95rem;
}
</style>
```

```
</head>
<body>
  <div class="panel">
    <h1>Canary Deployment</h1>
    <p>Instance is running updated build.</p>
    <div class="version">Version 2</div>
  </div>
</body>
</html>
EOF

systemctl restart nginx
```

## Step 6 - Creating Version 2 Target Group

### What was done:

A new target group `shloka-canary-task-v2-TG` was created and the v2 instance was attached.

### Why was it done:

This TG is used in the weighted ALB listener rule to route 10% traffic to the new build of the app.

### How was it done:

- TG type: instance-based
- Protocol: HTTP
- Health check: `/`
- Registered canary instance ( `server-test` )

The screenshot shows the AWS Lambda function configuration page. At the top, there are tabs for 'Overview', 'Code', 'Test', 'Logs', and 'Metrics'. Below these, the function name is 'shloka-canary-task-v2'. Under the 'Handler' section, it shows 'lambda.lambda\_handler' and 'Python 3.8'. The 'Memory limit' is set to 128 MB. The 'Timeout' is set to 10 seconds. The 'Environment' section lists variables: 'AWS\_LAMBDA\_FUNCTION\_NAME' (shloka-canary-task-v2), 'AWS\_LAMBDA\_FUNCTION\_MEMORY\_SIZE' (128), 'AWS\_LAMBDA\_FUNCTION\_TIMEOUT' (10), and 'AWS\_LAMBDA\_FUNCTION\_SOURCE' (Local file system). The 'Layers' section shows 'shloka-canary-task-layer' (version 1). The 'Triggers' section lists 'shloka-canary-task-v1' (EventBridge) and 'shloka-canary-task-v2' (EventBridge). The 'Logs' section shows log groups: '/aws/lambda/shloka-canary-task-v2' and '/aws/lambda/shloka-canary-task-v2/main'. The 'Actions' section includes 'Edit', 'Delete', 'Version history', and 'Logs'.

Healthy instance verification of TG v2

## Step 7 - Editing ALB Listener Rules for Canary Deployment

### What was done:

Weighted forwarding was configured so the ALB sends 90% traffic to v1 and 10% to v2.

### Why was it done:

To perform a safe canary deployment where only a fraction of real traffic hits the new version before full rollout.

### How was it done:

- Edited **default rule**

- Changed action:

- v1-TG → 90%
- v2-TG → 10%

**HTTP:80** Info

**Details**

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

**ProtocolPort**  
HTTP:80

**Load balancer**  
[shloka-canary-task-alb](#)

**Listener ARN**  
[arn:aws:elasticloadbalancing:us-west-2:739324485843:listener/app/shloka-canary-task-alb/2b9a5d1dd4f10d40/4d8008d07bc9ae0](#)

**Default actions**

- Forward to target group
  - [shloka-canary-task-v1-TG](#) 90 (90%)
  - [shloka-canary-task-v2-TG](#) 10 (10%)

**Target group stickiness:** Off

**Rules** Attributes Tags

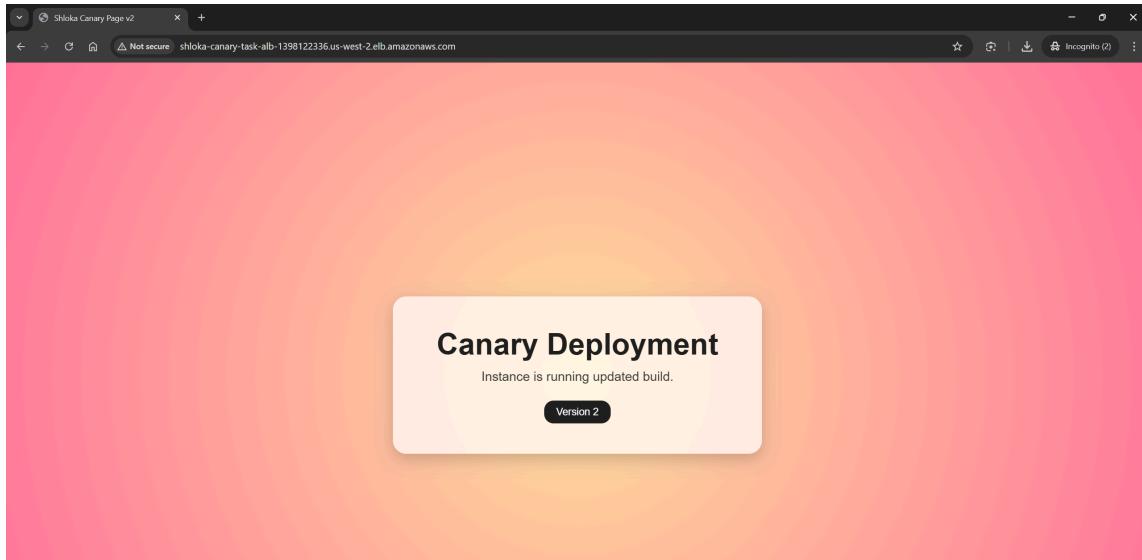
**Listener rules (1)** Info

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

| Filter rules | Priority       | Name tag | Conditions (If)          | Transforms | Actions (Then)  | ARN                 | Tags   | Actions                                     |
|--------------|----------------|----------|--------------------------|------------|---|---------------------|--------|---|
|              | Low            | Default  | If no other rule applies | -          | <ul style="list-style-type: none"> <li>• Forward to target group           <ul style="list-style-type: none"> <li><a href="#">shloka-canary-task-v1-TG</a> 90 (90%)</li> <li><a href="#">shloka-canary-task-v2-TG</a> 10 (10%)</li> </ul> </li> </ul> | <a href="#">ARN</a> | 0 tags | <a href="#">Edit</a> <a href="#">Delete</a> |
|              | Last (default) |          |                          |            |   |                     |        |   |

Rule limits: [Edit](#) [Actions](#) [Add rule](#)

Updated ALB listening rules



verifying canary deployment after multiple refreshes

## Step 8 - Setting Up S3 Bucket for ALB Access Logs

### **What was done:**

Created an S3 bucket + added necessary bucket policy for ALB to write logs.

### **Why was it done:**

To validate the canary setup by checking real distribution of requests across both target groups.

### **How was it done:**

- Created bucket: `shloka-alb-logs-canary-task`
- Added required bucket policy:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AWSLogDeliveryWrite",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::797873946194:root"  
      },  
      "Action": "s3:PutObject",  
      "Resource": "arn:aws:s3:::shloka-alb-logs-canary-task/AWSLogs/  
*",  
      "Condition": {}  
    },  
    {  
      "Sid": "AWSLogDeliveryAclCheck",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "delivery.logs.amazonaws.com"  
      },  
      "Action": "s3:GetBucketAcl",  
      "Resource": "arn:aws:s3:::shloka-alb-logs-canary-task"  
    }  
  ]  
}
```

- In ALB → **Attributes** → **Access Logs**

Added bucket → enabled logging.

## Step 9 - Validating Canary Traffic Distribution Using ALB Access Logs

### What was done:

Analyzed ALB access logs stored in S3 to confirm that traffic was successfully distributed between Version 1 (v1) and Version 2 (v2) target groups according to the 90/10 canary weighting.

### Why was it done:

A canary deployment must **prove** that:

- Only a controlled fraction of traffic reached the canary version
- Most traffic remained on the stable version

ALB access logs provide irrefutable backend evidence of request routing.

This serves as:

- Deployment validation
- Documentation for the assignment
- Safety confirmation before rolling out fully to v2

### How was it done:

- After enabling logging, generated traffic by refreshing ALB DNS repeatedly.
- Navigated to:

breadcrumb path and log file in s3

- Opened each **.log.gz** file → used S3 “Open” to downloaded locally.
- Inside each log entry, searched for v2:

```

1  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-69329078-78287db06b3c78e423f02
2  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-69329078-1ca4190a58301e585f597
3  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-69329079-648f19a23ff52a8a1818c
4  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907a-6890144869d697a5b992
5  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907a-5b1fccc81d36cc4b22a84
6  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/d09923a443172537 "Root=1-6932907c-279ace1621061000f9b
7  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907c-d35c4426bf719eb794b3
8  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v2-TG/d09923a443172537 "Root=1-6932907c-582f0bc5471b6afcf066e
9  "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907d-40a224346df9ff4c5248c2
10 "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v2-TG/d09923a443172537 "Root=1-6932907b-4913f50863d9898e7650
11 "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907b-7a2f6c751d928e1ee72
12 "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907b-7ae37b2f4fd2bfd750a37
13 "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907b-1db20f702b20f36613b
14 "iri/537.36" - - arn:aws:elasticloadbalancing:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c7e54e644a63 "Root=1-6932907c-69fc069b34fc21955f3b3
15

```

## Step 10 - Implementing the Rollback Plan (Reset Traffic to 100% v1)

### What was done:

Performed a full rollback by updating the ALB listener rules to send **100% traffic** to the stable Version 1 target group and **0% to Version 2**.

### Why was it done:

A rollback plan is crucial in production-style canary deployments.

If performance issues, bugs, or instability are detected in the canary build, teams must immediately shift all traffic back to the stable version.

Rollback ensures:

- Zero downtime
- No impact on end users
- Rapid recovery without any infrastructure teardown

**How was it done:**

Modified weighted forwarding:

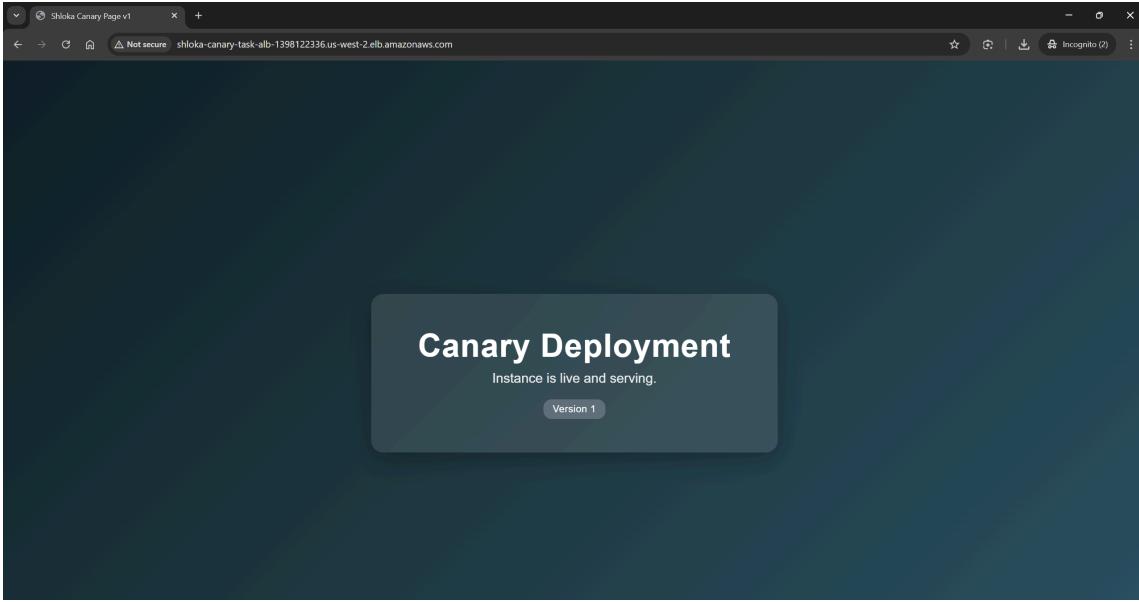
- Changed **v1-TG → 100%**
- Changed **v2-TG → 0%**
- Saved rule changes
- Waited a few seconds for ALB propagation
- Validated rollback:
  - Refreshed ALB DNS repeatedly
  - Verified all responses were Version 1
  - Checked ALB logs (optional) confirming all requests routed to v1

The screenshot shows the AWS Application Load Balancer (ALB) configuration interface. In the 'Default action' section, the 'Forward to target groups' option is selected. Under 'Routing action', there are three options: 'Forward to target groups' (selected), 'Redirect to URL', and 'Return fixed response'. In the 'Forward to target group' section, two target groups are listed:

| Target group             | Protocol | Weight | Percent | Action |
|--------------------------|----------|--------|---------|--------|
| shloka-canary-task-v1-TG | HTTP     | 100    | 100%    | Remove |
| shloka-canary-task-v2-TG | HTTP     | 0      | 0%      | Remove |

Below the target groups, there is a section for 'Target group stickiness' which is currently turned off. At the bottom, there is a 'Server-side tasks and status' section and a 'Save changes' button.

updated listening rules



showing v1 only. SUCCESSFULLY CHANGED Listening rules

```
739324-1.LOG x
C: > Users > Shloka > Downloads > 739324~1.LOG > 739324-1.LOG
| > v2 Aa ab, w* No results ↑ ↓ ≡ x
1 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299b7-4f5d481f41149e4e70de605d" "-.-" "0" 2025-12-05T08:37:11.120000Z
2 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299b8-45730bd559f6c8754648a11" "-.-" "0" 2025-12-05T08:37:12.692000Z
3 ng:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/d09923a443172537 "Root=1-693299d0-5062516a5ce07c1b5f656c98" "-.-" "0" 2025-12-05T08:37:36.543000Z
4 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d1-0c984766b980e487e6b3685" "-.-" "0" 2025-12-05T08:37:37.822000Z
5 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d2-6e8475fc0ed13b79734bf6f5" "-.-" "0" 2025-12-05T08:37:38.050000Z
6 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d2-68704b2667fe00e53cb4dbd0" "-.-" "0" 2025-12-05T08:37:38.596000Z
7 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d2-649547e92fa6e36683660bf" "-.-" "0" 2025-12-05T08:37:39.722000Z
8 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d2-17a6d398210ee0b7f8372d7" "-.-" "0" 2025-12-05T08:37:39.946000Z
9 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d3-0b637c54f2558actf59fd60f" "-.-" "0" 2025-12-05T08:37:39.185000Z
10 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d3-09f6ca0754b63d041456c3d" "-.-" "0" 2025-12-05T08:37:39.618000Z
11 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d3-3df97d267692110189d6f0a" "-.-" "0" 2025-12-05T08:37:39.800000Z
12 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d4-102872be4df1a76f5a553b07" "-.-" "0" 2025-12-05T08:37:40.157000Z
13 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d4-109b433103b549076adad50" "-.-" "0" 2025-12-05T08:37:40.460000Z
14 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d4-6a8cfa63f0a47444d0f08d236" "-.-" "0" 2025-12-05T08:37:40.758000Z
15 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d5-15dd59a62583d81a546c849" "-.-" "0" 2025-12-05T08:37:41.748000Z
16 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d6-039eff245fbcb31864299c7b" "-.-" "0" 2025-12-05T08:37:42.600000Z
17 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d7-1f91ca0871631a251c6ceae0" "-.-" "0" 2025-12-05T08:37:43.360000Z
18 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d8-170460371ed13f3d35be87fe" "-.-" "0" 2025-12-05T08:37:44.160000Z
19 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d8-748dd536f495ac47ffba0" "-.-" "0" 2025-12-05T08:37:44.956000Z
20 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d9-16d9b53d7d24ac252a986b1" "-.-" "0" 2025-12-05T08:37:45.610000Z
21 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d9-3e1401846aef3f331d424eb" "-.-" "0" 2025-12-05T08:37:46.274000Z
22 g:us-west-2:739324485843:targetgroup/shloka-canary-task-v1-TG/177c77e54e644a63 "Root=1-693299d9-3da02fb4d3998f1907dc4a39d" "-.-" "0" 2025-12-05T08:37:46.892000Z
```

updated logs after changes no v2