Civic Issue Reporter – Phase 1 Deployment (Domain Live)

This document records **everything done so far** to deploy the Civic Issue Reporter application and make it accessible via a custom domain. This is **Phase 1: Basic EC2 + Domain Deployment (No HTTPS yet)**.

---

1. Project Overview

Project Name: Civic Issue Reporter
Type: Web application (Flask-based)
Purpose: Allow users to report civic issues and track their status via a dashboard

---

2. AWS Account & Region Setup

- AWS account created/logged in
- Region selected (example: `ap-south-1`)

---

3. EC2 Instance Creation

3.1 Launch EC2 Instance

- Service: **Amazon EC2**
- Instance type: `t2.micro` (Free Tier)
- AMI: **Ubuntu 24.04 LTS**
- Key pair: Created/downloaded (`.pem` file)
- Security Group rules:
- SSH (22) → My IP
- HTTP (80) → 0.0.0.0/0
- Custom TCP (5000) → 0.0.0.0/0 (temporary)

3.2 Connect to EC2

```
ssh -i your-key.pem ubuntu@<EC2-PUBLIC-IP>
```

---

4. System Preparation on EC2

4.1 Update System

```
sudo apt update && sudo apt upgrade -y
```

4.2 Install Required Packages

```
sudo apt install python3 python3-venv python3-pip git -y
```

---

5. Application Setup

5.1 Clone Project Repository

```
git clone <your-github-repo-url>
cd civic-issue-reporter1
```

5.2 Create Virtual Environment

```
python3 -m venv venv
source venv/bin/activate
```

5.3 Install Dependencies

```
pip install -r requirements.txt
```

---

6. Running the Flask Application

6.1 Run Application Manually

```
python3 app.py
```

or (production-style with Gunicorn):

```
gunicorn --bind 0.0.0.0:5000 app:app
```

6.2 Verify Application

- Access in browser:

```
http://<EC2-PUBLIC-IP>:5000
```

- Dashboard loads successfully

---

7. Domain Purchase & DNS Configuration

7.1 Domain Purchased

- Domain: `civicissuereporter.in`
- Purchased from external registrar

7.2 Create Hosted Zone in Route 53

- Service: **AWS Route 53**
- Hosted Zone name: `civicissuereporter.in`
- Record type: Public Hosted Zone

7.3 Update Nameservers at Registrar

Copied Route 53 nameservers and updated them in domain registrar:

```
ns-411.awsdns-51.com
ns-XXXX.awsdns-XX.net
ns-XXXX.awsdns-XX.org
ns-XXXX.awsdns-XX.co.uk
```

---

8. DNS Record Configuration

8.1 Create A Record

- Record type: `A`
- Name: `civicissuereporter.in`
- Value: `<EC2-PUBLIC-IP>`
- TTL: 300

---

9. DNS Verification

9.1 Check DNS Propagation

```
dig civicissuereporter.in
```

Result: - Status: `NOERROR` - SOA record pointing to AWS Route 53 nameservers

---

10. Final Verification

10.1 Access Application via Domain

```
http://civicissuereporter.in
```

10.2 Observations

- Application loads correctly
- Dashboard visible
- Domain resolves properly
- Browser shows **"Not Secure"** (HTTP only)

---

11. Current Architecture (Phase 1)

```
User Browser
     ↓
Domain (Route 53)
     ↓
EC2 Public IP
     ↓
Gunicorn (Port 5000)
     ↓
Flask Application
```

---

12. What Is Completed

- EC2-based deployment
- Python virtual environment
- Flask app running with Gunicorn
- Custom domain mapped successfully

- End-to-end connectivity verified

---

13. Known Limitations (Intentional for Phase 1)

- No HTTPS (SSL not configured)
- Port 5000 exposed publicly
- No Load Balancer
- No Auto Scaling
- No CI/CD
- No monitoring/logging

---

14. Next Planned Phases

Phase 2: HTTPS + Load Balancer (ACM + ALB)
Phase 3: Dockerization
Phase 4: ECS / Production Deployment
Phase 5: CI/CD with GitHub Actions

---

15. Summary

This phase validates: - Cloud fundamentals - EC2 deployment - DNS & Route 53 understanding - Real-world domain hosting

This is a **solid foundation**, but **not production-ready yet**.

Phase 2 will convert this into a **secure, professional deployment**.