Sevabot AWS Deployment Guide - Updated for Port 8001 & Dynamic URLs

Prerequisites

- AWS Account with IAM user `SevabotFullAccess`
- ECR Repository: `908924926461.dkr.ecr.ap-south-1.amazonaws.com/vcd-tech/sevabot-mvp-gradio`
- Existing EC2 instance (30GB) with .pem file access
- GitHub repository with your Sevabot code

Step 1: Project Structure Setup

Create this directory structure in your project:

```
sevabot/
├── main.py
├── requirements.txt
├── config.py (updated)
├── constants.py (updated)
├── auth.py (updated)
├── Dockerfile
├── .dockerignore
├── .env.example
├── start_local.sh
├── .github/
│   └── workflows/
│       └── deploy.yml
├── nginx/
│   └── sevabot.conf
└── (your other project files)
```

Step 2: Prepare Your EC2 Instance

SSH into your existing EC2 instance:

bash

```
ssh -i your-key.pem ubuntu@your-ec2-public-ip
```

Install Docker and dependencies:

bash

```
# Update system
sudo apt-get update

# Install Docker
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce
sudo usermod -aG docker ubuntu

# Install other dependencies
sudo apt-get install -y curl jq git nginx

# Create data directories for persistence
mkdir -p /home/ubuntu/sevabot_data/user_documents
mkdir -p /home/ubuntu/sevabot_data/rag_index

# Reboot to apply docker group changes
sudo reboot
```

After reboot, SSH back in and verify Docker works:

```
bash

docker --version
```

Set up GitHub Actions Runner:

```
bash
```

```
# Create runner directory
mkdir actions-runner && cd actions-runner

# Download latest runner
curl -o actions-runner-linux-x64-2.311.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.311.0/actior

# Extract
tar xzf ./actions-runner-linux-x64-2.311.0.tar.gz

# Configure (get token from GitHub: Settings > Actions > Runners > New self-hosted runner)
./config.sh --url https://github.com/YOUR_USERNAME/YOUR_REPO --token YOUR_TOKEN

# Install as service
sudo ./svc.sh install
sudo ./svc.sh start

# Verify it's running
sudo ./svc.sh status
```

Step 3: GitHub Repository Setup

Add GitHub Secrets:

Go to your GitHub repository → Settings → Secrets and variables → Actions

Add these secrets (replace with your actual EC2 DNS):

- `AWS_ACCESS_KEY_ID`: Your SevabotFullAccess user access key
- `AWS_SECRET_ACCESS_KEY`: Your SevabotFullAccess user secret access key
- `SUPABASE_URL`: Your Supabase project URL
- `SUPABASE_KEY`: Your Supabase anon key
- `SUPABASE_SERVICE_ROLE_KEY`: Your Supabase service role key
- `GOOGLE_CLIENT_ID`: Your Google OAuth client ID
- `GOOGLE_CLIENT_SECRET`: Your Google OAuth client secret
- `COOKIE_SECRET`: Generate with `openssl rand -base64 32`
- `COOKIE_NAME`: `sevabot_session`
- `OPENAI_API_KEY`: Your OpenAI API key (use OPENAI_API_KEY, not OPENAI_KEY)
- `REDIRECT_URI`: `http://ec2-XX-XXX-XX-XXX.ap-south-1.compute.amazonaws.com/auth/callback`
- `APP_HOST`: `http://ec2-XX-XXX-XX-XXX.ap-south-1.compute.amazonaws.com`
- `ALLOWED_DOMAIN`: `sadhguru.org`

Step 4: Update Google OAuth Settings

In your Google Cloud Console:

1. Go to APIs & Services → Credentials
2. Edit your OAuth 2.0 client
3. Add to **Authorized redirect URIs** (NOT JavaScript origins):
   - `http://ec2-XX-XXX-XX-XXX.ap-south-1.compute.amazonaws.com/auth/callback`
   - Keep your localhost URI for development: `http://localhost:8001/auth/callback`

Step 5: Update EC2 Security Groups

Ensure these ports are open:

- Port 22 (SSH) - for your access
- Port 80 (HTTP) - for user access to your app
- Port 443 (HTTPS) - if you add SSL later

Remove port 8001 from security group if you added it - it's not needed.

Step 6: Local Development Setup

For local development, create a `.env` file from the template:

```bash
cp .env.example .env
# Edit .env with your local values (localhost:8001 URLs)
```

Start locally:

```bash
chmod +x start_local.sh
./start_local.sh
```

Step 7: Deploy Your Application

Push your code to trigger deployment:

```bash
git add .
git commit -m "Add production deployment configuration with port 8001"
git push origin main
```

Monitor the deployment:

1. Check GitHub Actions tab for workflow progress
2. SSH into your EC2 instance to monitor:

```bash
# Check container status
docker ps

# Check container logs
docker logs sevabot-container

# Check nginx status
sudo systemctl status nginx

# Check if app is responding
curl http://localhost:8001/health
```

Step 8: Access Your Application

Your application will be available at:

- `http://YOUR_EC2_PUBLIC_IP` (port 80 via Nginx)
- `http://YOUR_EC2_PUBLIC_DNS` (port 80 via Nginx)

Step 9: Daily DNS Update Process

When your EC2 instance restarts and gets a new DNS/IP:

1. **Get new EC2 public DNS** from AWS console
2. **Update GitHub secrets**:
   - `REDIRECT_URI`: `http://NEW_EC2_DNS/auth/callback`
   - `APP_HOST`: `http://NEW_EC2_DNS`
3. **Update Google OAuth** redirect URI in Google Cloud Console
4. **Redeploy** (push any commit or manually trigger workflow)

Step 10: SSL Setup (Optional but Recommended)

For production with a domain name:

```bash
```

```
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Get SSL certificate (replace with your domain)
sudo certbot --nginx -d your-domain.com

# Auto-renewal
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet
```

Troubleshooting

Check Docker container:

```bash
docker logs sevabot-container
docker exec -it sevabot-container /bin/bash
curl http://localhost:8001/health
```

Check Nginx:

```bash
sudo nginx -t
sudo systemctl status nginx
sudo journalctl -u nginx -f
```

Check GitHub Actions Runner:

```bash
cd ~/actions-runner
sudo ./svc.sh status
```

Common Issues:

1. **Container not starting**: Check environment variables in GitHub secrets
2. **OAuth redirect errors**: Verify Google OAuth redirect URIs match your current EC2 DNS
3. **Port 8001 connection refused**: Container might not be running, check `docker logs`
4. **Nginx 502 errors**: App container not responding on port 8001
5. **File upload issues**: Check nginx client_max_body_size setting

Debug Commands:

```bash
# Test internal app connection
curl http://localhost:8001/

# Test Nginx proxy
curl http://localhost/

# Check ports
netstat -tulpn | grep :8001
netstat -tulpn | grep :80

# Check Docker networks
docker network ls
docker inspect sevabot-container
```

File Persistence

The deployment uses Docker volumes to persist:

- User documents: `/home/ubuntu/sevabot_data/user_documents`
- RAG index: `/home/ubuntu/sevabot_data/rag_index`

These directories survive container restarts and updates.

Updates

To update your application:

1. Push changes to your main branch
2. GitHub Actions automatically builds and deploys
3. Zero-downtime deployment with container replacement
4. Data persists across deployments