

DevOps Documentation

Deployment, Infrastructure and Operations Guide

ChourangiHealth Video Consultation Platform

Version 1.0 | February 2026

1. Infrastructure Overview

1.1 Architecture Diagram

Browser (Client) connects via HTTPS to Nginx on EC2. Nginx routes traffic to OpenVidu Server, KMS, and FastAPI Backend. COTURN handles WebRTC media relay when direct connection fails. KMS handles all media streams between participants.

- EC2 Instance: c6a.xlarge, Ubuntu 24.04, eu-west-2 (London)
- Public IP: 18.130.149.212
- Private IP: 172.31.33.5
- Domain: chourangi.duckdns.org
- SSL: Let's Encrypt auto-provisioned by OpenVidu
- Containers: Docker managed via docker-compose

2. Complete Deployment Procedure

2.1 Full Fresh Deployment Checklist

Follow these steps in order for a clean deployment from scratch:

- Step 1: Launch EC2 c6a.xlarge with Ubuntu 24.04
- Step 2: Configure Security Group with all required ports (22, 80, 443, 3478, 40000-65535)
- Step 3: Register domain at duckdns.org pointing to EC2 public IP
- Step 4: SSH into EC2 and install OpenVidu using official script
- Step 5: Edit /opt/openvidu/.env with domain, email, and secret
- Step 6: Edit /opt/openvidu/docker-compose.yml to add KMS and COTURN fixes
- Step 7: Create /opt/openvidu/custom-nginx-locations/frontend.conf
- Step 8: Start OpenVidu with ./openvidu start
- Step 9: Clone backend repo and create .env file
- Step 10: Install backend dependencies and start uvicorn
- Step 11: Clone frontend repo, build, and copy to custom-layout
- Step 12: Test with two devices on different networks

2.2 docker-compose.yml KMS Section

Full kms service configuration in /opt/openvidu/docker-compose.yml:

```
kms:
  image: ${KMS_IMAGE:-kurento/kurento-media-server:7.3.0}
  restart: always
  network_mode: host
  environment:
    - KMS_EXTERNAL_IPV4=18.130.149.212
    - KMS_NETWORK_INTERFACES=ens5
    - KMS_ICE_TCP=0
    - KMS_MIN_PORT=40000
    - KMS_MAX_PORT=57000
```

3. Routine Deployment Commands

3.1 Deploy Backend Update

```
cd /home/ubuntu/test-1/backend
git fetch origin master
git reset --hard origin/master
kill -f uvicorn
nohup uvicorn app.main:app --host 0.0.0.0 --port 8000 > /tmp/backend.log 2>&1 &
sleep 3 && curl http://localhost:8000/health
```

3.2 Deploy Frontend Update

```
cd /home/ubuntu/test-1/frontend
git fetch origin master
git reset --hard origin/master
npm run build
cp -r build/* /opt/openvidu/custom-layout/
```

3.3 Restart OpenVidu

```
cd /opt/openvidu
./openvidu stop
./openvidu start > /tmp/openvidu.log 2>&1 &
```

NOTE: OpenVidu takes 60-90 seconds to fully start. Wait before testing.

3.4 Restart Only KMS Container

```
cd /opt/openvidu
docker-compose up -d --force-recreate kms
sleep 15
docker logs openvidu-kms-1 --tail 20
```

3.5 Full System Restart Command

```
cd /opt/openvidu && ./openvidu stop && ./openvidu start > /tmp/openvidu.log 2>&1 & sleep 90 && cd /home/ubuntu/test-1/backend && kill -f uvicorn && nohup uvicorn app.main:app --host 0.0.0.0 --port 8000 > /tmp/backend.log 2>&1 & sleep 3 && curl http://localhost:8000/health
```

4. Monitoring and Logs

4.1 Check Running Status

```
# Check backend is running
curl http://localhost:8000/health

# Check all Docker containers
docker ps -a

# Check OpenVidu server logs
cd /opt/openvidu && ./openvidu logs
```

4.2 Log File Locations

- Backend logs: /tmp/backend.log
- OpenVidu startup logs: /tmp/openvidu.log
- KMS logs: docker logs openvidu-kms-1
- COTURN logs: docker logs openvidu-coturn-1
- Nginx logs: docker logs openvidu-nginx-1
- OpenVidu server logs: docker logs openvidu-openvidu-server-1

4.3 Useful Diagnostic Commands

```
# Verify COTURN is listening on port 3478
ss -ulnp | grep 3478

# Check KMS config was applied
docker exec openvidu-kms-1 cat /etc/kurento/modules/kurento/WebRtcEndpoint.conf.ini | grep -v '^;:'

# Check KMS environment variables
docker exec openvidu-kms-1 env | grep KMS

# Test OpenVidu API directly
curl -u OPENVIDUAPP:openvidu-v2-32 http://localhost:5443/openvidu/api/sessions

# View backend logs live
tail -f /tmp/backend.log
```

5. Git Workflow

5.1 Local Development to Production Flow

- Step 1: Make code changes locally
- Step 2: Test locally with `npm start` and `uvicorn --reload`
- Step 3: Test between two devices using local IP (192.168.1.62)
- Step 4: Switch `API_BASE_URL` back to production URL in `api.js`
- Step 5: Commit and push to GitHub master branch
- Step 6: SSH to EC2 and pull latest code
- Step 7: Build frontend and restart backend
- Step 8: Test on production URL

5.2 GitHub Repositories

- Backend: https://github.com/AshuuuPatil/openvidu_videocall_microservice_backend
- Frontend: https://github.com/AshuuuPatil/openvidu_videocall_microservice_frontend

5.3 Git Best Practices for This Project

- Never commit `.env` files (they contain secrets)
- Never commit `__pycache__` or `*.pyc` files (add to `.gitignore`)
- Never commit `node_modules` or build directories
- Always use `git reset --hard origin/master` on EC2 to avoid merge conflicts
- Use `git clean -fd` after reset to remove any untracked files

5.4 EC2 GitHub Authentication

The frontend repository on EC2 requires a GitHub username and password (personal access token) when pulling. This is because it was cloned with HTTPS. For automation, configure SSH key authentication or use a GitHub personal access token stored in git credential cache.

6. File Paths Quick Reference

6.1 EC2 Server

- OpenVidu config: `/opt/openvidu/.env`
- OpenVidu docker-compose: `/opt/openvidu/docker-compose.yml`
- OpenVidu management: `/opt/openvidu/openvidu`
- Nginx custom locations: `/opt/openvidu/custom-nginx-locations/frontend.conf`
- Frontend files served: `/opt/openvidu/custom-layout/`
- Recordings: `/opt/openvidu/recordings/`
- Backend code: `/home/ubuntu/test-1/backend/`
- Frontend code: `/home/ubuntu/test-1/frontend/`
- Backend logs: `/tmp/backend.log`

6.2 Local Development (Windows)

- Backend: `D:\call.chourangihealth\2) POC\1)Code\openvidu-fastapi-poc\`
- Frontend: `D:\call.chourangihealth\2) POC\1)Code\openvidu-frontend-poc\`

7. Credentials Reference

NOTE: Store these securely. Do not share publicly or commit to git.

7.1 Application

- Doctor: username doctor, password doctor123
- Patient: username patient, password patient123

7.2 OpenVidu Admin

- Username: OPENVIDUAPP
- Password: openvidu-v2-32
- Dashboard URL: <https://chourangi.duckdns.org/dashboard>

7.3 EC2 SSH

- User: ubuntu
- Host: 18.130.149.212
- Authentication: PEM key file

7.4 DuckDNS

- Domain: chourangi.duckdns.org
- Admin: <https://www.duckdns.org>

7.5 AWS

- EC2 Region: eu-west-2 (London)
- Instance ID: available in AWS Console

8. Known Issues and Workarounds

8.1 Frontend GitHub Pull Requires Password

The frontend was cloned with HTTPS so every git pull asks for GitHub username and password. Workaround: Enter credentials manually each time. Permanent fix: Set up SSH key authentication or store credentials with `git config --global credential.helper store`.

8.2 Backend Not Surviving EC2 Restart

The unicorn process runs in background with `nohup` but does not automatically restart after an EC2 reboot. Workaround: SSH in and restart manually after reboot. Permanent fix: Create a systemd service file for unicorn so it starts automatically on boot.

8.3 OpenVidu Recording Requires Active Session

The recording start API returns a 500 error if called when no session is active or no publisher is connected. Always ensure both participants have joined and video is publishing before clicking Record.

8.4 Same Network Testing Fails

Testing with two devices on the same WiFi network (e.g. both on 192.168.1.x) may fail due to NAT hairpin issues. The KMS on EC2 cannot route media back to local IP addresses on the same LAN. Always test with one device on WiFi and one on mobile data (4G/5G) for reliable cross-network testing.

End of DevOps Documentation