Streaming QoS Lab

Installation and User Guide

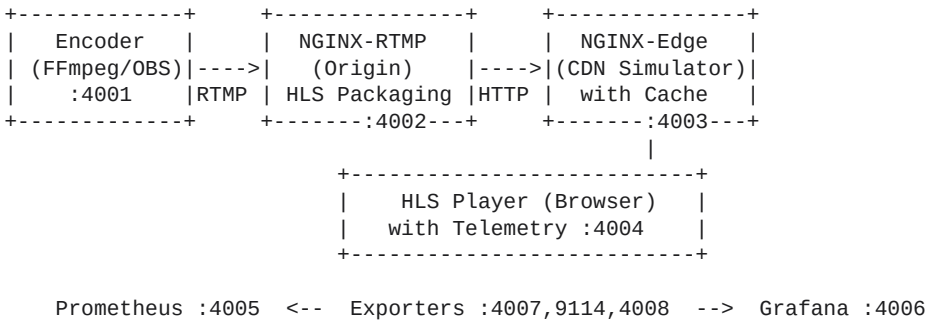
A complete lab environment simulating Apple-like streaming infrastructure with Origin, Edge CDN, HLS packaging, and QoS monitoring using Prometheus and Grafana.

Version	1.1
Date	January 2026
Platform	Linux (Ubuntu 22.04+)
Requirements	Docker, Docker Compose, FFmpeg
Ports	4001-4008, 9114

Table of Contents

1. Architecture Overview

The lab simulates professional video streaming infrastructure similar to Apple TV+, Netflix, or other major streaming services.



Components

Component	Port	Description
NGINX-RTMP	4001/4002	Origin - receives RTMP, outputs HLS
NGINX-Edge	4003	CDN simulator with caching (host network)
Web Player	4004	HLS player with real-time metrics
Prometheus	4005	Metrics database
Grafana	4006	Dashboards (admin/streaming123)
Origin Exporter	4007	NGINX metrics for Origin
Node Exporter	4008	System metrics (CPU, memory, network)
Edge Exporter	9114	NGINX metrics for Edge (host network)

Data Flow

1. Encoder sends RTMP to Origin (:4001)
2. Origin packages RTMP into HLS (.m3u8 + .ts)
3. Edge caches HLS from Origin
4. Players fetch HLS from Edge (:4003)
5. Prometheus scrapes metrics from all exporters
6. Grafana visualizes Origin vs Edge performance

2. Prerequisites

Component	Minimum	Recommended
CPU	2 cores	4+ cores
RAM	4 GB	8+ GB
Disk	10 GB	20+ GB
OS	Ubuntu 22.04	Ubuntu 24.04

Install Required Software

```
# Docker
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER
# Log out and back in

# Docker Compose (usually included with Docker)
sudo apt install docker-compose-plugin

# FFmpeg
sudo apt install ffmpeg
```

3. Installation

Step 1: Extract the Package

```
tar -xzvf streaming-lab.tar.gz
cd streaming-lab
```

Step 2: Run Setup Script (REQUIRED)

The setup script configures your server's IP address in all config files. This is required because some components use host networking.

```
chmod +x scripts/*.sh
./scripts/setup.sh
```

The script will auto-detect your IP and update: nginx-edge/nginx.conf, prometheus/prometheus.yml, docker-compose.yml, and grafana datasource.

Step 3: Start Services

```
docker-compose up -d

# Wait for services to initialize
sleep 30

# Check status
docker-compose ps
```

Step 4: Verify

```
# All containers should be Up
docker-compose ps

# Check Prometheus targets (all should be UP)
# Open http://YOUR_IP:4005/targets
```

4. Quick Start

Start Test Stream

```
# Option 1: Use the test script
./scripts/test-stream.sh

# Option 2: FFmpeg manually
ffmpeg -re \
  -f lavfi -i "testsrc=size=1280x720:rate=30" \
  -f lavfi -i "sine=frequency=440" \
  -c:v libx264 -preset ultrafast -b:v 2000k \
  -c:a aac -f flv rtmp://YOUR_IP:4001/live/test
```

Watch the Stream

```
# Web Player (recommended)
http://YOUR_IP:4004

# Enter HLS URL in player:
http://YOUR_IP:4003/hls/test.m3u8

# Or use VLC:
vlc http://YOUR_IP:4003/hls/test.m3u8
```

Open Grafana Dashboard

```
URL: http://YOUR_IP:4006
Username: admin
Password: streaming123

Go to: Dashboards > Streaming Lab > Streaming QoS Overview
```

Tip: Use Chrome or Firefox for full player metrics. Safari uses native HLS which limits visible metrics.

5. Using the Lab

CDN Cache Behavior

The Edge server adds X-Cache-Status headers to show cache behavior:

Status	Meaning
`MISS`	Not cached, fetched from origin
`HIT`	Served from edge cache
`EXPIRED`	Cached but expired, refreshed
`STALE`	Serving old content while refreshing

```
# Test cache behavior
curl -I http://YOUR_IP:4003/hls/test.m3u8 | grep X-Cache
# X-Cache-Status: MISS (playlist always fresh)

curl -I http://YOUR_IP:4003/hls/test-1.ts | grep X-Cache
# First request: MISS
# Second request: HIT
```

Understanding Origin vs Edge

In the Grafana dashboard, compare Origin and Edge metrics:

- Request Rate: Edge should have more requests than Origin (cache hits)
- Connections: Edge handles clients, Origin only handles cache misses
- Good cache efficiency: Edge 100 req/s, Origin 10 req/s = 90% hit ratio

Live Streaming Cache Behavior

For live streaming, you'll see mostly MISS on .ts segments because each segment is new. The value of the CDN is that:

- First viewer causes MISS (fetches from origin)
- All subsequent viewers get HIT (served from cache)
- With 1000 viewers: 1 MISS + 999 HITs = 99.9% efficiency

6. Monitoring with Grafana

The pre-configured dashboard shows:

- Origin Status and Edge Status (UP/DOWN)
- Origin vs Edge Active Connections
- Request Rate comparison (Origin vs Edge)
- System CPU and Memory Usage
- Network Throughput

Prometheus Queries

```
# Request rate comparison
rate(nginx_http_requests_total{instance="origin"}[1m])
rate(nginx_http_requests_total{instance="edge"}[1m])

# Active connections
nginx_connections_active{instance="origin"}
nginx_connections_active{instance="edge"}

# CPU usage
100 - (avg(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)

# Memory usage
(1 - (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes)) * 100
```

7. Command Reference

```
# Start/Stop
docker-compose up -d
docker-compose down
docker-compose down -v  # Also remove volumes

# Logs
docker-compose logs -f
docker-compose logs -f nginx-rtmp
docker-compose logs -f nginx-edge

# Restart specific service
docker-compose restart nginx-edge
docker-compose restart prometheus

# Status
docker-compose ps
./scripts/info.sh

# Verify cache
curl -I http://YOUR_IP:4003/hls/test.m3u8 | grep X-Cache
```

8. Troubleshooting

Setup Not Run

If you see HOST_IP in config files or Prometheus shows targets DOWN:

```
# Run setup script
./scripts/setup.sh

# Restart all services
docker-compose down
docker-compose up -d
```

Docker Network Issues

If containers can't communicate (context deadline exceeded):

```
sudo systemctl restart docker
docker-compose down
docker-compose up -d
```

Stream Not Playing

```
# Check RTMP receiving
curl http://localhost:4002/stat

# Check HLS exists
curl http://localhost:4002/hls/test.m3u8

# Check Edge
curl http://localhost:4003/hls/test.m3u8

# Check logs
docker-compose logs nginx-rtmp | tail -20
```

Grafana No Data

```
# 1. Check Prometheus targets
http://YOUR_IP:4005/targets
# All should be UP

# 2. Check Grafana datasource
# Settings > Data Sources > Prometheus > Save & Test
# URL should be: http://YOUR_IP:4005
```

Port Already in Use

```
# Find what's using the port
sudo lsof -i :4003
sudo netstat -tlnp | grep 4003
```

Good luck with your streaming journey!