Vcse-cd2.0环境搭建

缩写

CTES：CDN Transcode E2E Sample

OVC:

CDN:

# CDN Transcode E2E Sample Getting Started Guide

[TOC]

This document describes how to run the CDN Transcode E2E Sample (abbr as CTES) step by step. Please reference \*\*OVC\_CDN\_Transcode\_E2E\_Sample\_RA.md\*\* to understand the CTES reference architecture design to understand how CTES works.

The CTES provides CDN transcode services, and this guide just shows how to use the services in a simplest and typical way which can be scaled out to more complex environment. E.g.: in this guide, the docker images for transcoder server and cdn edge server are hosted on the same physical server, in real case, they can be hosted on differented servers located in different places in the CDN network.

# Live Broadcasting

## Environment Setup

Below diagram shows how the CTES nodes interconnect with each other and the network topology used in this guide.

```

+------------------+-----------------+-----+

| Transcode Server | CDN Edge Server | |

+------------------+ | [Ubuntu docker] | [Ubuntu docker] | | +------------------+

| Streaming Server | | 192.168.31.31 | 192.168.31.32 | ... | | Client |

| PC1 10.67.117.70 |----->|------------------+-----------------+-----|---->| PC2 10.67.117.80 |

| FFmpeg | | E3/E5: Ubuntu l8.04 10.67.116.179 | | VLC |

+------------------+ +------------------------------------------+ +------------------+

```

### Install docker on E3/E5 host

```sh

sudo apt-get install docker.io

```

### Setup docker proxy on E3/E5 host

```

sudo mkdir -p /etc/systemd/system/docker.service.d

printf "[Service]\nEnvironment=\"HTTPS\_PROXY=$https\_proxy\" \"NO\_PROXY=$no\_proxy\"\n" | sudo tee /etc/systemd/system/docker.service.d/proxy.conf

sudo systemctl daemon-reload

sudo systemctl restart docker

```

### Build docker images

On E5 Server, run below command to build docker images:

``` sh

mkdir build

cd build

cmake ..

cd xcode-server/ffmpeg

make

ctest

cd cdn-server/nginx+rtmp

make

ctest

```

On E3 Server, run below command to build docker images:

``` sh

mkdir build

cd build

cmake ..

cd xcode-server/ffmpeg+vaapi+qsv

make

ctest

cd cdn-server/nginx+rtmp

make

ctest

```

### Create docker network

Run below command on E3/E5 server:

```

docker network create -d bridge --subnet 192.168.31.0/24 --gateway 192.168.31.1 my\_bridge

```

### Run transcoder server docker instance

Run below command on E5 server:

```

docker run -it --network=my\_bridge --ip 192.168.31.31 --name xcoder xeon-ubuntu1804-ffmpeg:1.0 /bin/bash

Run:

ffmpeg -i rtmp://10.67.117.70/live/nature -c:v libsvt\_hevc -f flv rtmp://192.168.31.32/hls/nature -c:v libsvt\_hevc -f flv rtmp://192.168.31.32/dash/nature -abr\_pipeline

```

Run below command on E3 server:

```

docker run -it --device=/dev/dri:/dev/dri --network=my\_bridge --ip 192.168.31.31 --name xcoder xeon-ubuntu1804-ffmpeg:1.0 /bin/bash

Run:

ffmpeg -hwaccel vaapi -hwaccel\_device /dev/dri/renderD128 -hwaccel\_output\_format vaapi -i rtmp://10.67.117.70/live/nature -c:v h264\_vaapi -f flv rtmp://192.168.31.32/hls/nature -c:v h264\_vaapi -f flv rtmp://192.168.31.32/dash/nature -abr\_pipeline

```

### Run CDN server docker instance

Run below command on E3/E5 server:

```

docker run -it -p 80:80 --network=my\_bridge --ip 192.168.31.32 --name nginx xeon-ubuntu1804-nginx-rtmp:1.0 /bin/bash

Run:

nginx &

```

### Install FFmpeg on Streaming Server

Run below command on streaming server:

```

sudo apt-get install -y ffmpeg

```

### Install VLC on client

Run below command on client:

```

sudo apt-get install -y vlc

```

## Sample Execution

### Streaming

Run below command on streaming server:

```

ffmpeg -re -stream\_loop 500 -i Nature.mp4 -c:v copy -an -f flv rtmp://10.67.117.70/live/nature

```

### Live play

Run below command on client:

```

vlc http://10.67.116.179/hls/nature/index.m3u8

vlc http://10.67.116.179/dash/nature/index.mpd

```

# Video on Demand

## Environment Setup

Below diagram shows how the CTES nodes interconnect with each other and the network topology used in this guide.

```

+------------------+-----------------+-----+

| Transcode Server | CDN Edge Server | |

| [Ubuntu docker] | [Ubuntu docker] | |

| 192.168.31.31 | 192.168.31.32 | ... |

+------------------+-----------------+-----+

| Zookeeper | Kafka Server |

+------------------+ | [Ubuntu docker] | [Ubuntu docker] | +------------------+

| Streaming Server | | 192.168.31.29 | 192.168.31.30 | | Client |

| PC1 10.67.117.70 |----->|------------------+-----------------+-----|---->| PC2 10.67.117.80 |

| FFmpeg | | E3/E5: Ubuntu l8.04 10.67.116.179 | | Browser |

+------------------+ +------------------------------------------+ +------------------+

```

### Install docker on E3/E5 host

```sh

sudo apt-get install docker.io

```

### Setup docker proxy on E3/E5 host

```

sudo mkdir -p /etc/systemd/system/docker.service.d

printf "[Service]\nEnvironment=\"HTTPS\_PROXY=$https\_proxy\" \"NO\_PROXY=$no\_proxy\"\n" | sudo tee /etc/systemd/system/docker.service.d/proxy.conf

sudo systemctl daemon-reload

sudo systemctl restart docker

```

### Build docker images

On E5 Server, run below command to build docker images:

``` sh

mkdir build

cd build

cmake ..

cd xcode-server/ffmpeg

make

ctest

cd cdn-server/nginx+rtmp

make

ctest

```

On E3 Server, run below command to build docker images:

``` sh

mkdir build

cd build

cmake ..

cd xcode-server/ffmpeg+vaapi+qsv

make

ctest

cd cdn-server/nginx+rtmp

make

ctest

```

### Create docker network

Run below command on E3/E5 server:

```

docker network create -d bridge --subnet 192.168.31.0/24 --gateway 192.168.31.1 my\_bridge

```

### Run zookeeper server docker instance

Run below command on E3/E5 server:

```

docker pull confluentinc/cp-zookeeper

docker run -d --name zookeeper --env ZOOKEEPER\_SERVER\_ID=1 --env ZOOKEEPER\_CLIENT\_PORT=2181 --env ZOOKEEPER\_TICK\_TIME=2000 --restart=always --network=my\_bridge --ip 192.168.31.29 -it confluentinc/cp-zookeeper

```

### Run kafka server docker instance

Run below command on E3/E5 server:

```

docker pull confluentinc/cp-kafka

docker run -d --name kafka --link zookeeper --env KAFKA\_BROKER\_ID=1 --env KAFKA\_ZOOKEEPER\_CONNECT=zookeeper:2181 --env KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://kafka:9092,PLAINTEXT\_HOST://localhost:29092 --env KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP=PLAINTEXT:PLAINTEXT,PLAINTEXT\_HOST:PLAINTEXT --env KAFKA\_INTER\_BROKER\_LISTENER\_NAME=PLAINTEXT --env KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR=1 --env KAFKA\_DEFAULT\_REPLICATION\_FACTOR=1 --env KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE=true --env CONFLUENT\_SUPPORT\_METRICS\_ENABLE=0 --restart=always --network=my\_bridge --ip 192.168.31.30 -it confluentinc/cp-kafka

```

### Run transcoder server docker instance

Run below command on E5 server:

```

docker run -it --network=my\_bridge --ip 192.168.31.31 --name xcoder -v /var/www/dash:/var/www/dash -v /var/www/hls:/var/www/hls -v /var/www/archive:/var/www/archive xeon-ubuntu1804-ffmpeg:1.0 /bin/bash

Run:

python3 main.py

```

Run below command on E3 server:

```

docker run -it --device=/dev/dri:/dev/dri --network=my\_bridge --ip 192.168.31.31 --name xcoder xeon-ubuntu1804-ffmpeg:1.0 /bin/bash

Run:

python3 main.py

```

### Run CDN server docker instance

Run below command on E3/E5 server:

```

docker run -it -p 80:80 --network=my\_bridge --ip 192.168.31.32 --name nginx -v /var/www/dash:/var/www/dash -v /var/www/hls:/var/www/hls -v /var/www/archive:/var/www/archive xeon-ubuntu1804-nginx-rtmp:1.0 /bin/bash

Run:

nginx &

python3 main.py

```

### Configure input streaming parameter on transcoder server docker instance and CDN server docker instance

Modify below parameter at config.ini:

```

[mode] # input streaming mode

srcMode=local # live - the input is remote rtmp streaming, local - the input is a local stream

[path] # local stream path or remote desktop IP.

srcPath=/var/www/archive

```

### Install FFmpeg and Nginx on Streaming Server

If input streaming mode is live, run below command on streaming server:

```

sudo apt-get install -y ffmpeg nginx

```

Modify location / element to the following at nginx.conf, then copy streams to the path you defined, such as /data/www/file.

```

location / {

root /data/www/file;

autoindex on;

autoindex\_exact\_size off;

autoindex\_localtime on;

}

```

## Sample Execution

### Streaming

If input streaming mode is live, run below command on streaming server:

```

ffmpeg -re -stream\_loop 500 -i /data/www/file/Nature.mp4 -c:v copy -an -f flv rtmp://10.67.117.70/live/Nature.mp4

```

### Live play

Access URL:http://10.67.116.179/ with browser.