Pau de Batlle 242171
Maria Gómez 218011
Mark Zharikov 218390

Equips i Sistemes de Video
P4 - CDN & docker for encoding

1.-) Download Docker for command line or for Desktop and install it. The first you are going to create a container which contains the software FFmpeg and you are going to be able to run that container passing FFmpeg commands.

Un cop tenim docker desktop instal·lat seguim la següent guia per crear un contenidor i una imatge amb ffmpeg instal·lat:

Hem seguit aquesta guia per crear una aplicació en python:
https://docs.docker.com/language/python/containerize/

i en el dockerfile afegim aquesta línia per instal·lar ffmpeg i poder fer servir comandes desde el cli:

```
RUN apt-get -y update && apt-get -y upgrade && apt-get install -y
--no-install-recommends ffmpeg
```

finalment un cop ho tenim tot muntat fem build i up per pujar el contenidor i la imatge i poder-la fer servir i aquesta comanda per empaquetar un minut de video .mp4 a:

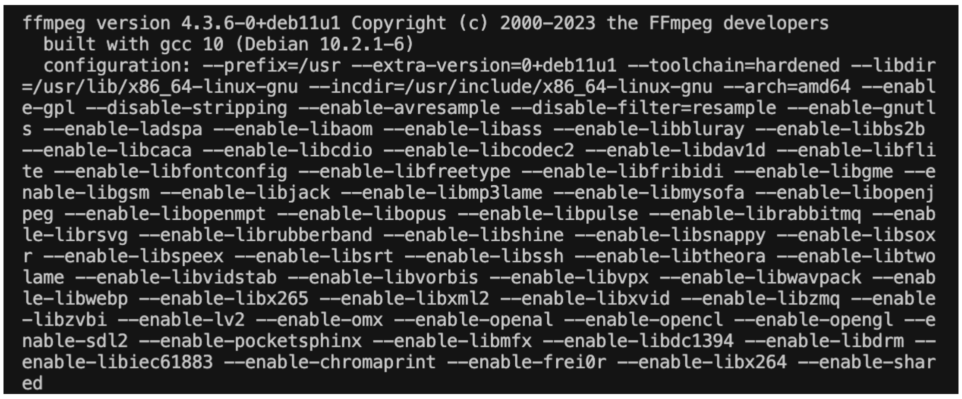
- MP4 container with HLS - Video .h264 AVC, audio AAC

```
docker-compose exec server ffmpeg -i /app/BigBuckBunny.mp4 \
  -c:v libx264 -preset veryfast -tune film -profile:v baseline -level:v 3.0 \
  -c:a aac -strict experimental -b:a 128k \
  -hls_time 10 -hls_playlist_type vod -hls_segment_filename "/tmp/output_%03d.ts"
/tmp/output.m3u8
```

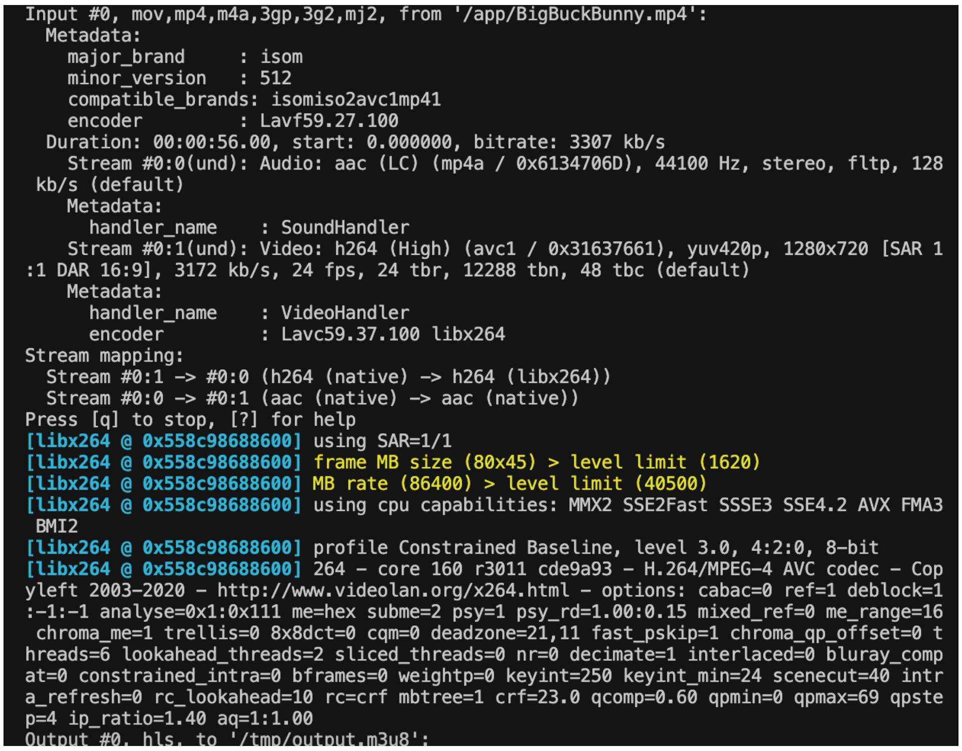
- **docker-compose exec server**: executar la comanda dins del contenidor "server"
- **-i /app/BigBuckBunny.mp4**: input video
- **-c:v libx264 -preset veryfast -tune film -profile:v baseline -level:v 3.0**: video encoder presets (h264)
- **-c:a aac -strict experimental -b:a 128k**: audio encoder presets (aac)
- **-hls_time 10 -hls_playlist_type vod -hls_segment_filename "/tmp/output_%03d.ts" /tmp/output.m3u8**: generar HLS (HTTP Live Streaming) llista i segments.
- **-hls_time 10**: duració de cada segment (10s)
- **-hls_playlist_type vod**: especifica el tipus de llista (Video on Demand)
- **-hls_segment_filename "/tmp/output_%03d.ts" /tmp/output.m3u8**: especifica el output

Pau de Batlle 242171
Maria Gómez 218011
Mark Zharikov 218390

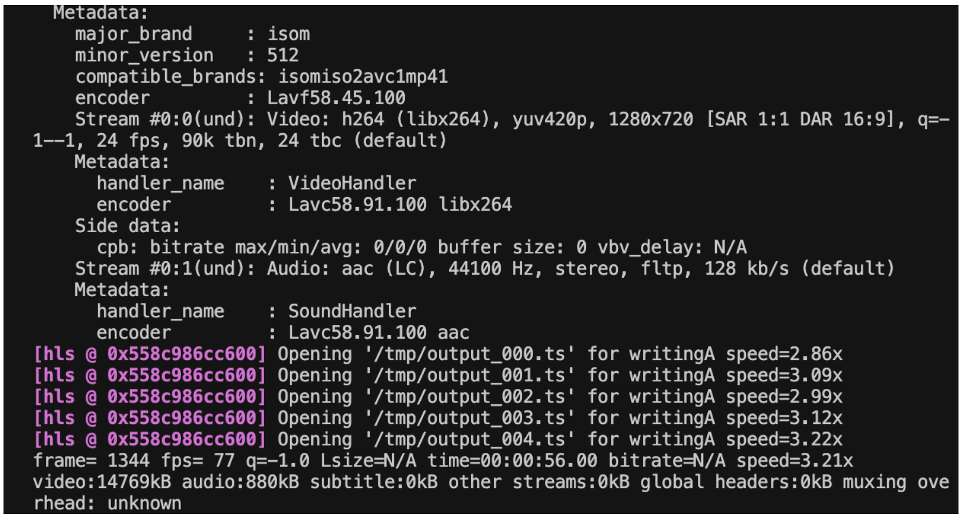
cli output:



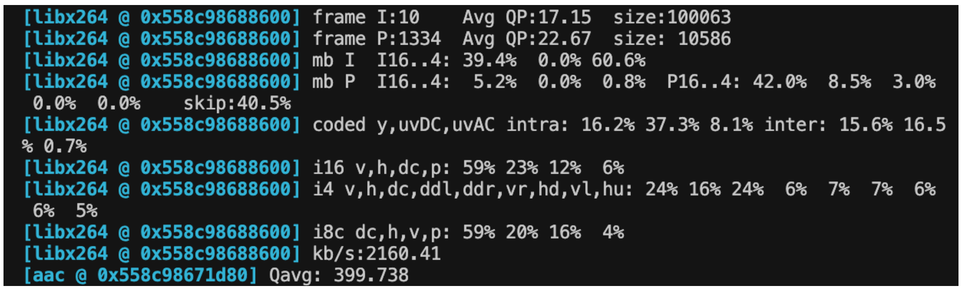
```
ffmpeg version 4.3.6-0+deb11u1 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 10 (Debian 10.2.1-6)
  configuration: --prefix=/usr --extra-version=0+deb11u1 --toolchain=hardened --libdir
=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enabl
e-gpl --disable-stripping --enable-avresample --disable-filter=resample --enable-gnutl
s --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b
--enable-libcaca --enable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libfli
te --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --e
nable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenj
peg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbitmq --enab
le-librsvg --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsox
r --enable-libspeex --enable-libsrt --enable-libssh --enable-libtheora --enable-libtwo
lame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwavpack --enab
le-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable
-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --e
nable-sdl2 --enable-pocketsphinx --enable-libmfx --enable-libdc1394 --enable-libdrm --
enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shar
ed
```



```
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '/app/BigBuckBunny.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf59.27.100
  Duration: 00:00:56.00, start: 0.000000, bitrate: 3307 kb/s
    Stream #0:0(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 128
 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
    Stream #0:1(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1280x720 [SAR 1
:1 DAR 16:9], 3172 kb/s, 24 fps, 24 tbr, 12288 tbn, 48 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc59.37.100 libx264
Stream mapping:
  Stream #0:1 -> #0:0 (h264 (native) -> h264 (libx264))
  Stream #0:0 -> #0:1 (aac (native) -> aac (native))
Press [q] to stop, [?] for help
[libx264 @ 0x558c98688600] using SAR=1/1
[libx264 @ 0x558c98688600] frame MB size (80x45) > level limit (1620)
[libx264 @ 0x558c98688600] MB rate (86400) > level limit (40500)
[libx264 @ 0x558c98688600] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3
 BMI2
[libx264 @ 0x558c98688600] profile Constrained Baseline, level 3.0, 4:2:0, 8-bit
[libx264 @ 0x558c98688600] 264 - core 160 r3011 cde9a93 - H.264/MPEG-4 AVC codec - Cop
yleft 2003-2020 - http://www.videolan.org/x264.html - options: cabac=0 ref=1 deblock=1
:-1:-1 analyse=0x1:0x111 me=hex subme=2 psy=1 psy_rd=1.00:0.15 mixed_ref=0 me_range=16
 chroma_me=1 trellis=0 8x8dct=0 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=0 t
hreads=6 lookahead_threads=2 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_comp
at=0 constrained_intra=0 bframes=0 weightp=0 keyint=250 keyint_min=24 scenecut=40 intr
a_refresh=0 rc_lookahead=10 rc=crf mbtree=1 crf=23.0 qcomp=0.60 qpmin=0 qpmax=69 qpste
p=4 ip_ratio=1.40 aq=1:1.00
Output #0, hls, to '/tmp/output.m3u8':
```



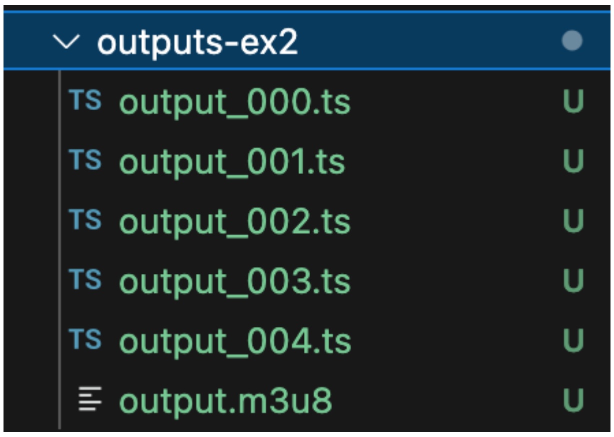
```
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.45.100
    Stream #0:0(und): Video: h264 (libx264), yuv420p, 1280x720 [SAR 1:1 DAR 16:9], q=-
1--1, 24 fps, 90k tbn, 24 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc58.91.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: N/A
    Stream #0:1(und): Audio: aac (LC), 44100 Hz, stereo, fltp, 128 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
      encoder         : Lavc58.91.100 aac
[hls @ 0x558c986cc600] Opening '/tmp/output_000.ts' for writingA speed=2.86x
[hls @ 0x558c986cc600] Opening '/tmp/output_001.ts' for writingA speed=3.09x
[hls @ 0x558c986cc600] Opening '/tmp/output_002.ts' for writingA speed=2.99x
[hls @ 0x558c986cc600] Opening '/tmp/output_003.ts' for writingA speed=3.12x
[hls @ 0x558c986cc600] Opening '/tmp/output_004.ts' for writingA speed=3.22x
frame= 1344 fps= 77 q=-1.0 Lsize=N/A time=00:00:56.00 bitrate=N/A speed=3.21x
video:14769kB audio:880kB subtitle:0kB other streams:0kB global headers:0kB muxing ove
rhead: unknown
```



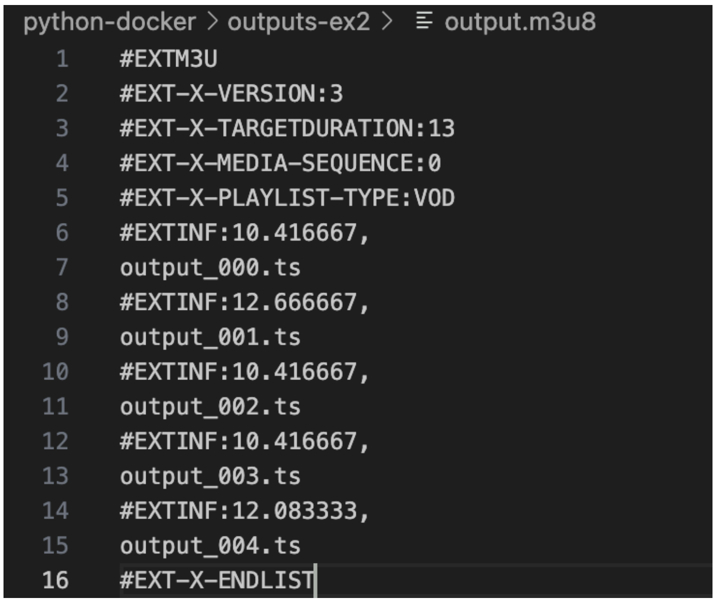
```
[libx264 @ 0x558c98688600] frame I:10    Avg QP:17.15  size:100063
[libx264 @ 0x558c98688600] frame P:1334  Avg QP:22.67  size: 10586
[libx264 @ 0x558c98688600] mb I  I16..4: 39.4%  0.0% 60.6%
[libx264 @ 0x558c98688600] mb P  I16..4:  5.2%  0.0%  0.8%  P16..4: 42.0%  8.5%  3.0%
 0.0%  0.0%    skip:40.5%
[libx264 @ 0x558c98688600] coded y,uvDC,uvAC intra: 16.2% 37.3% 8.1% inter: 15.6% 16.5
% 0.7%
[libx264 @ 0x558c98688600] i16 v,h,dc,p: 59% 23% 12%  6%
[libx264 @ 0x558c98688600] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 24% 16% 24%  6%  7%  7%  6%
 6%  5%
[libx264 @ 0x558c98688600] i8c dc,h,v,p: 59% 20% 16%  4%
[libx264 @ 0x558c98688600] kb/s:2160.41
[aac @ 0x558c98671d80] Qavg: 399.738
```

Pau de Batlle 242171
Maria Gómez 218011
Mark Zharikov 218390

Finalment copiem els arxius generats al contenidor a la nostra màquina local per obtenir:



```
outputs-ex2
  output_000.ts    U
  output_001.ts    U
  output_002.ts    U
  output_003.ts    U
  output_004.ts    U
  output.m3u8      U
```



python-docker > outputs-ex2 > output.m3u8

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:13
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:10.416667,
output_000.ts
#EXTINF:12.666667,
output_001.ts
#EXTINF:10.416667,
output_002.ts
#EXTINF:10.416667,
output_003.ts
#EXTINF:12.083333,
output_004.ts
#EXT-X-ENDLIST
```

3.-) Now that you know how to 'Docker', search for the Bento4 software. Put it inside a Docker, and try to apply a DRM for the Previous packaged file.

Per aplicar DRM necessitem instalar Bento4 al nostre contenidor. Per conseguir-ho adaptem el dockerfile d'algú que ja ho havia fet per encaixar-ho al nostre.
https://github.com/alfg/docker-bento4/tree/master

Un cop ben adaptat i funcionant per aplicar DRM necessitem un .mp4 fragmentat així que fem servir la següent comanda:

Command: docker exec 582b8f793e41 mp4fragment /app/BigBuckBunny.mp4 /tmp/fragmented_BigBuckBunny.mp4

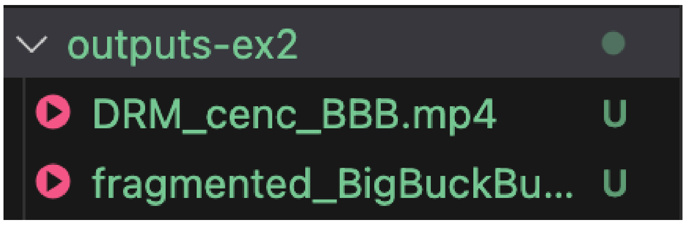
Output: found regular I-frame interval: 804 frames (at 24.000 frames per second)

I finalment després de moltes hores aconseguim aplicar DRM fent servir una comanda modificada treta de la documentació de Bento4:

docker exec 582b8f793e41 mp4encrypt --method MPEG-CENC --key 1:a0a1a2a3a4a5a6a7a8a9aaabacadaeaf:0123456789abcdef --property 1:KID:121a0fca0f1b475b8910297fa8e0a07e --key 2:a0a1a2a3a4a5a6a7a8a9aaabacadaeaf:aaaaaaaabbbbbbbb --property 2:KID:121a0fca0f1b475b8910297fa8e0a07e /tmp/fragmented_BigBuckBunny.mp4 /tmp/DRM_cenc_BBB.mp4
https://www.bento4.com/developers/dash/encryption_and_drm/

Pau de Batlle 242171
Maria Gómez 218011
Mark Zharikov 218390

Ara només falta copiar els arxius generats fora del contenidor i obtenim:



```
outputs-ex2
  DRM_cenc_BBB.mp4          U
  fragmented_BigBuckBu...   U
```

The docker image is in:
https://hub.docker.com/repository/docker/paudebatlle/p4_docker

The code is in:
https://github.com/paudBatlle/UNI_work/tree/main/Video_Equipment_and_Systems/P4_Docker

4.-) Now you're a master using Docker! Now try to follow-up this tutorial, take screenshots and make your comments:
https://github.com/leandromoreira/cdn-up-and-running.git
Some remarks: The way it works it's changing git versions (git checkout); this means when you move forward/back it will change files; if you get lost, try to repeat quick the git commands