# Contents

# Abstract

This research explores the performance of FFmpeg-based HTTP Live Streaming (HLS) running in a Virtual Machine (VM) environment and on Docker containers. A 4K video file, named drone.mp4, was transcoded into 1080p resolution at 1024 Mbps and 2048 Mbps bitrates. The resource usage was observed, along with the time taken for the transcoding task to complete. Video complexity was also measured using the SI-TI tool in terms of structural and temporal complexities. Results show that Docker performs better, with a 35% decrease in transcoding time and lower CPU usage compared to the VM. The findings also highlight the impact of hardware specifications and settings on video processing efficiency.

# Introduction

Video streaming is the majority of the internet traffic worldwide. HLS refers to a set of protocols used for video streaming at different qualities and segments. FFmpeg is a powerful open-source tool for transcoding HLS content.

The Windows Subsystem for Linux allows native Linux distributions to be installed and run inside the Windows operating system. This feature provides developers with an easy way to use Linux tools, such as FFmpeg. The research examines the effectiveness of FFmpeg in an HLS transcoding scenario on Ubuntu running under WSL on various host machine configurations. The aim is to demonstrate the impact of the host machine's performance on video processing operations performed under WSL.

# Methodology

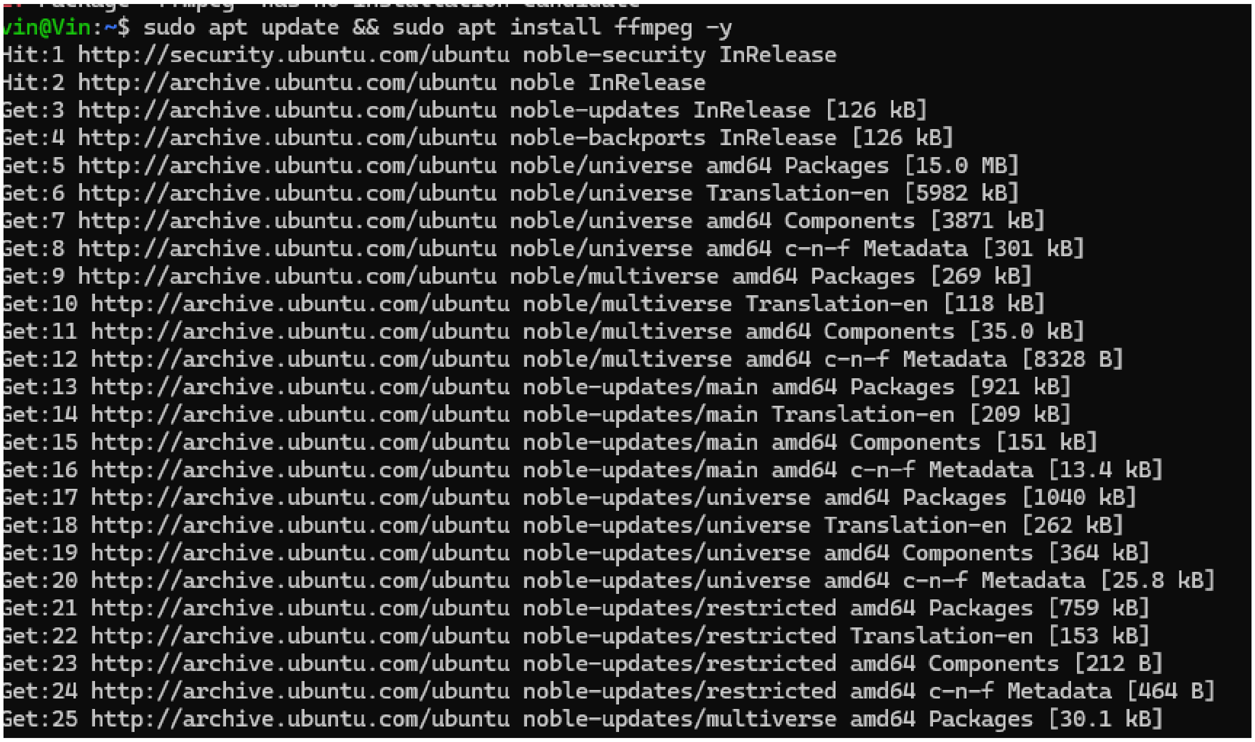
## VM-based FFmpeg Deployment

1. **Ubuntu/WSL Setup**:

Set up Ubuntu on Windows 11 I already have Ubuntu 22.04 LTS installed and running on my Windows 11 machine using WSL. To install FFmpeg, I opened the Ubuntu terminal and ran the following commands[1]:

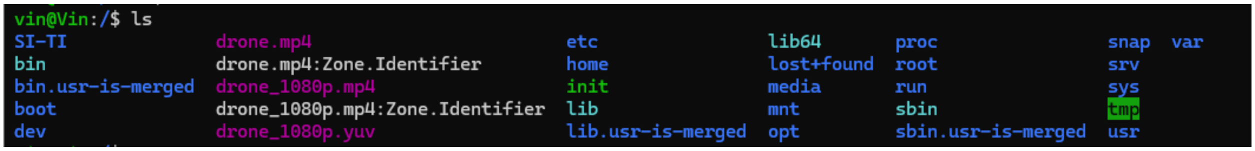
```
sudo apt update
```

```
sudo apt install ffmpeg
```



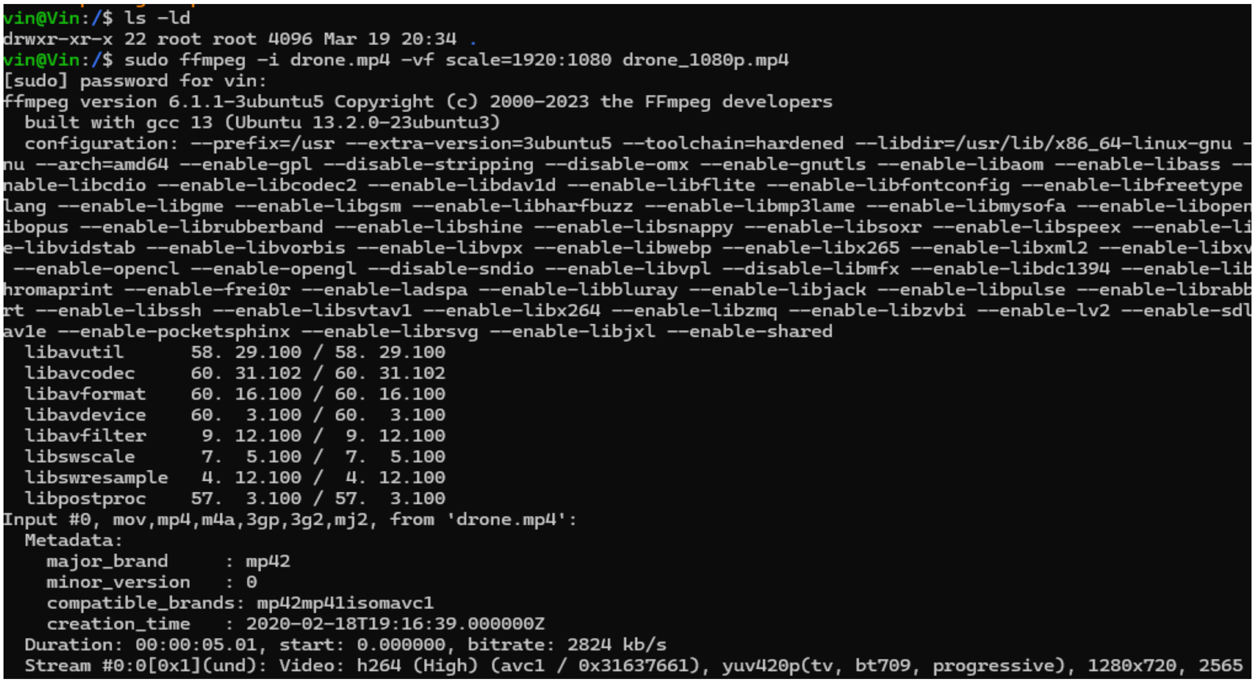
```
vin@Vin:~$ sudo apt update && sudo apt install ffmpeg -y
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [921 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [209 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [13.4 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1040 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [262 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [25.8 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [759 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [153 kB]
Get:23 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:24 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [464 B]
Get:25 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [30.1 kB]
```

2. **Input Video Selection**
   I downloaded a 4K video named "drone.mp4" from pexels.com and placed it in the Linux filesystem under the \\wsl.localhost\Ubuntu-24.04 directory, which is accessible from both Windows and Ubuntu.



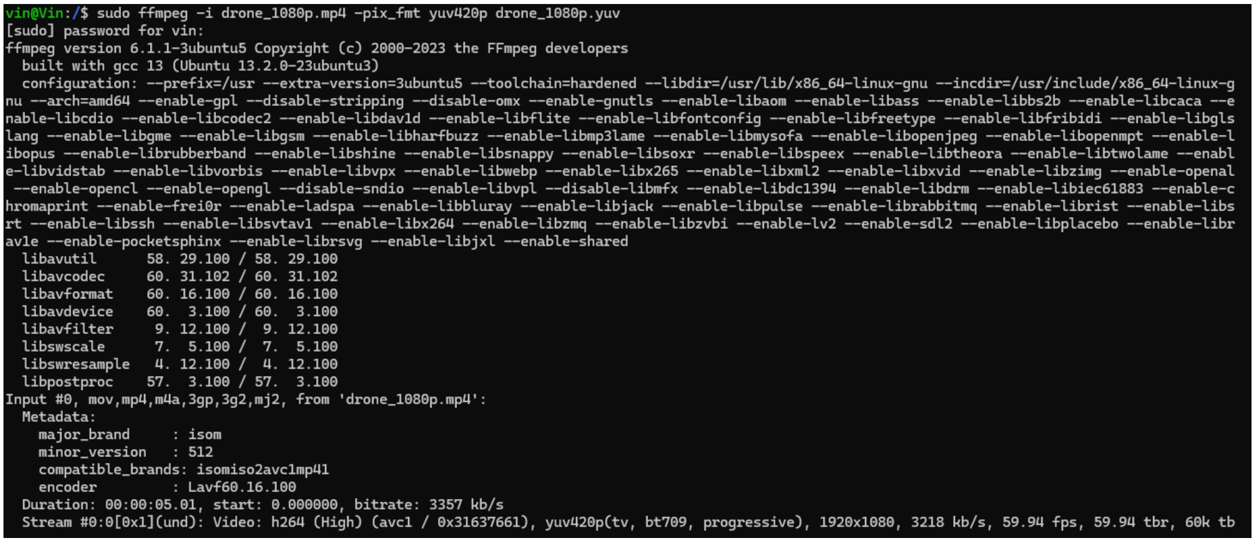
```
vin@Vin:/$ ls
SI-TI               drone.mp4                      etc               lib64       proc                snap  var
bin                 drone.mp4:Zone.Identifier      home              lost+found  root                srv
bin.usr-is-merged   drone_1080p.mp4                init              media       run                 sys
boot                drone_1080p.mp4:Zone.Identifier lib              mnt         sbin                tmp
dev                 drone_1080p.yuv                lib.usr-is-merged opt         sbin.usr-is-merged  usr
```

3. Scale/Resize Video To resize the video to 1080p resolution, I used the following FFmpeg command in the Ubuntu terminal. The resulting output video is drone_1080p.mp4
   Sudo ffmpeg -I drone.mp4 -vf scale=1920:1080 drone_1080p.mp4



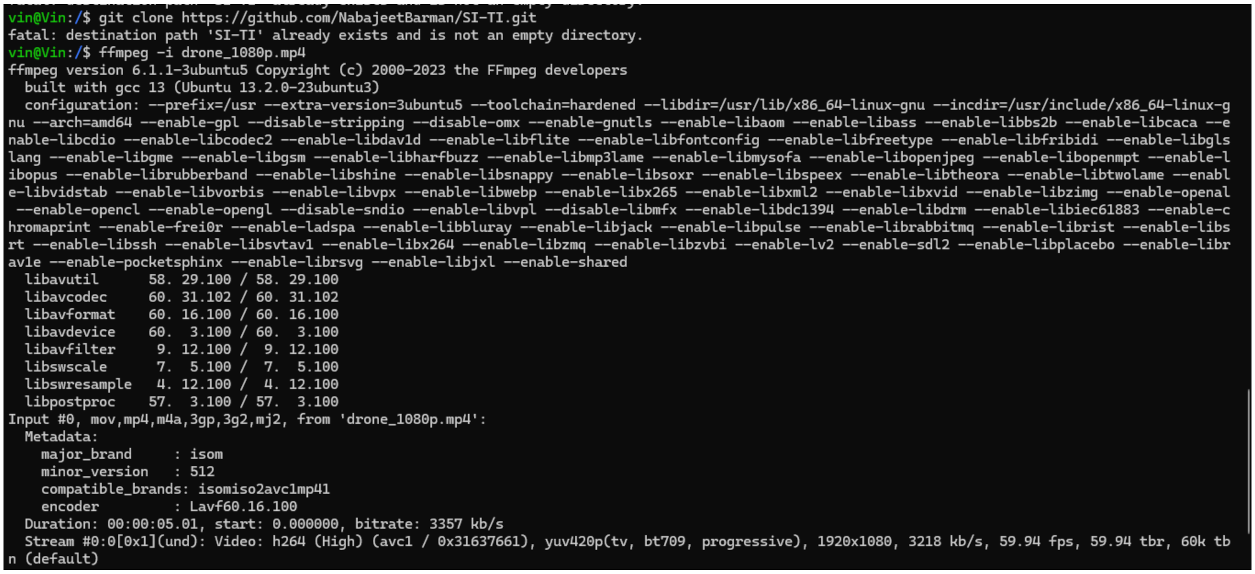
```
vin@Vin:/$ ls -ld
drwxr-xr-x 22 root root 4096 Mar 19 20:34 .
vin@Vin:/$ sudo ffmpeg -i drone.mp4 -vf scale=1920:1080 drone_1080p.mp4
[sudo] password for vin:
ffmpeg version 6.1.1-3ubuntu5 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Ubuntu 13.2.0-23ubuntu3)
  configuration: --prefix=/usr --extra-version=3ubuntu5 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu -
nu --arch=amd64 --enable-gpl --disable-stripping --disable-omx --enable-gnutls --enable-libaom --enable-libass --
nable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype
lang --enable-libgme --enable-libgsm --enable-libharfbuzz --enable-libmp3lame --enable-libmysofa --enable-libopen
ibopus --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-li
e-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxv
 --enable-opencl --enable-opengl --disable-sndio --enable-libvpl --disable-libmfx --enable-libdc1394 --enable-lib
hromaprint --enable-frei0r --enable-ladspa --enable-libbluray --enable-libjack --enable-libpulse --enable-librabb
rt --enable-libssh --enable-libsvtav1 --enable-libx264 --enable-libzmq --enable-libzvbi --enable-lv2 --enable-sdl
av1e --enable-pocketsphinx --enable-librsvg --enable-libjxl --enable-shared
  libavutil      58. 29.100 / 58. 29.100
  libavcodec     60. 31.102 / 60. 31.102
  libavformat    60. 16.100 / 60. 16.100
  libavdevice    60.  3.100 / 60.  3.100
  libavfilter     9. 12.100 /  9. 12.100
  libswscale      7.  5.100 /  7.  5.100
  libswresample   4. 12.100 /  4. 12.100
  libpostproc    57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'drone.mp4':
  Metadata:
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: mp42mp41isomavc1
    creation_time   : 2020-02-18T19:16:39.000000Z
  Duration: 00:00:05.01, start: 0.000000, bitrate: 2824 kb/s
  Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709, progressive), 1280x720, 2565
```

4. **I Cloned the GitHub repository to my machine.**
   *git clone https://github.com/NabajeetBarman/SI-TI.git*
   *cd SI-TI*
   Since the video is in MP4 format, I needed to convert it to YUV 4:2:0 format.
   I used *ffmpeg -i drone_1080.mp4 -pix_fmt yuv420p drone_1080.yuv* to do this conversion.



```
vin@Vin:/$ sudo ffmpeg -i drone_1080p.mp4 -pix_fmt yuv420p drone_1080p.yuv
[sudo] password for vin:
ffmpeg version 6.1.1-3ubuntu5 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Ubuntu 13.2.0-23ubuntu3)
  configuration: --prefix=/usr --extra-version=3ubuntu5 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-g
nu --arch=amd64 --enable-gpl --disable-stripping --disable-omx --enable-gnutls --enable-libaom --enable-libass --enable-libbs2b --enable-libcaca --e
nable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgls
lang --enable-libgme --enable-libgsm --enable-libharfbuzz --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-l
ibopus --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libtheora --enable-libtwolame --enabl
e-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-openal
 --enable-opencl --enable-opengl --disable-sndio --enable-libvpl --disable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-c
hromaprint --enable-frei0r --enable-ladspa --enable-libbluray --enable-libjack --enable-libpulse --enable-librabbitmq --enable-librist --enable-libs
rt --enable-libssh --enable-libsvtav1 --enable-libx264 --enable-libzmq --enable-libzvbi --enable-lv2 --enable-sdl2 --enable-libplacebo --enable-libr
av1e --enable-pocketsphinx --enable-librsvg --enable-libjxl --enable-shared
  libavutil      58. 29.100 / 58. 29.100
  libavcodec     60. 31.102 / 60. 31.102
  libavformat    60. 16.100 / 60. 16.100
  libavdevice    60.  3.100 / 60.  3.100
  libavfilter     9. 12.100 /  9. 12.100
  libswscale      7.  5.100 /  7.  5.100
  libswresample   4. 12.100 /  4. 12.100
  libpostproc    57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'drone_1080p.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf60.16.100
  Duration: 00:00:05.01, start: 0.000000, bitrate: 3357 kb/s
  Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709, progressive), 1920x1080, 3218 kb/s, 59.94 fps, 59.94 tbr, 60k tb
```

Determined the bit depth of your YUV video. Based on the repository, the scripts support both 8-bit and 10-bit YUV videos.



```
vin@Vin:/$ git clone https://github.com/NabajeetBarman/SI-TI.git
fatal: destination path 'SI-TI' already exists and is not an empty directory.
vin@Vin:/$ ffmpeg -i drone_1080p.mp4
ffmpeg version 6.1.1-3ubuntu5 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Ubuntu 13.2.0-23ubuntu3)
  configuration: --prefix=/usr --extra-version=3ubuntu5 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-g
nu --arch=amd64 --enable-gpl --disable-stripping --disable-omx --enable-gnutls --enable-libaom --enable-libass --enable-libbs2b --enable-libcaca --e
nable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgls
lang --enable-libgme --enable-libgsm --enable-libharfbuzz --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-l
ibopus --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libtheora --enable-libtwolame --enabl
e-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-openal
 --enable-opencl --enable-opengl --disable-sndio --enable-libvpl --disable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-c
hromaprint --enable-frei0r --enable-ladspa --enable-libbluray --enable-libjack --enable-libpulse --enable-librabbitmq --enable-librist --enable-libs
rt --enable-libssh --enable-libsvtav1 --enable-libx264 --enable-libzmq --enable-libzvbi --enable-lv2 --enable-sdl2 --enable-libplacebo --enable-libr
av1e --enable-pocketsphinx --enable-librsvg --enable-libjxl --enable-shared
  libavutil      58. 29.100 / 58. 29.100
  libavcodec     60. 31.102 / 60. 31.102
  libavformat    60. 16.100 / 60. 16.100
  libavdevice    60.  3.100 / 60.  3.100
  libavfilter     9. 12.100 /  9. 12.100
  libswscale      7.  5.100 /  7.  5.100
  libswresample   4. 12.100 /  4. 12.100
  libpostproc    57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'drone_1080p.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf60.16.100
  Duration: 00:00:05.01, start: 0.000000, bitrate: 3357 kb/s
  Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709, progressive), 1920x1080, 3218 kb/s, 59.94 fps, 59.94 tbr, 60k tb
n (default)
```

Based on the FFmpeg output provided, the  video file "drone_1080p.mp4" had the following properties:

Stream #0:00x1: Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709, progressive), 1920x1080, 3218 kb/s, 59.94 fps, 59.94 tbr, 60k tbn (default)

The important information here are:

- Codec: H.264 (High profile)
- Pixel format: yuv420p
- Resolution: 1920x1080
- Frame rate: 59.94 fps

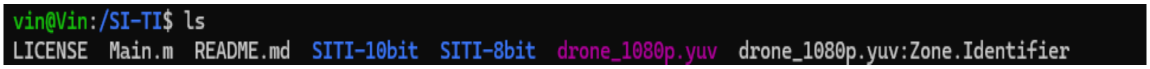
The pixel format "yuv420p" indicates that the video is 8-bit YUV with 4:2:0 chroma subsampling. The "p" at the end stands for "planar" format.

In YUV420p, the "420" represents the chroma subsampling ratio, where the U and V components have half the horizontal and vertical resolution of the Y component. The "p" means that the Y, U, and V components are stored in separate planes.

Since there is no "p10" or "p10le" in the pixel format, it confirms that the video is 8-bit and not 10-bit.

So, based on the FFmpeg output, I concluded that the "drone_1080p.mp4" video is an 8-bit YUV video with 4:2:0 chroma subsampling.

I Moved the converted YUV video file (drone_1080.yuv) to the "8-bit" folder in the cloned repository.



```
vin@Vin:/SI-TI$ ls
LICENSE  Main.m  README.md  SITI-10bit  SITI-8bit  drone_1080p.yuv  drone_1080p.yuv:Zone.Identifier
```

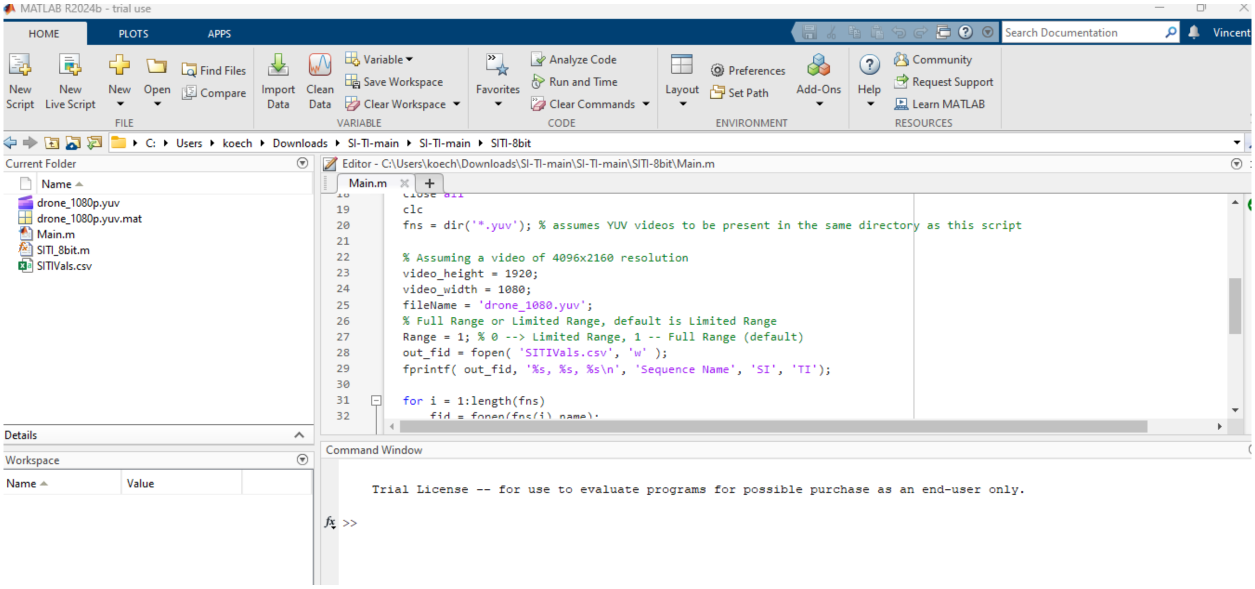
Using MATLAB, I navigated to the "8-bit" folder in the MATLAB Current Folder window. I then opened the Main.m script in the MATLAB editor. I modify the Main.m script to include your video resolution and filename.

For this instance, my video resolution is 1920x1080, update the following lines:

width = 1920;

height = 1080;

fileName = 'drone_1080.yuv';



C: ▸ Users ▸ koech ▸ Downloads ▸ SI-TI-main ▸ SI-TI-main ▸ SITI-8bit

```
drone_1080p.yuv
drone_1080p.yuv.mat
Main.m
SITI_8bit.m
SITIVals.csv
```

Editor - C:\Users\koech\Downloads\SI-TI-main\SI-TI-main\SITI-8bit\Main.m

Main.m

```
close all
clc
fns = dir('*.yuv'); % assumes YUV videos to be present in the same directory as this script

% Assuming a video of 4096x2160 resolution
video_height = 1920;
video_width = 1080;
fileName = 'drone_1080.yuv';
% Full Range or Limited Range, default is Limited Range
Range = 1; % 0 --> Limited Range, 1 -- Full Range (default)
out_fid = fopen( 'SITIVals.csv', 'w' );
fprintf( out_fid, '%s, %s, %s\n', 'Sequence Name', 'SI', 'TI');

for i = 1:length(fns)
    fid = fopen(fns(i).name);
```

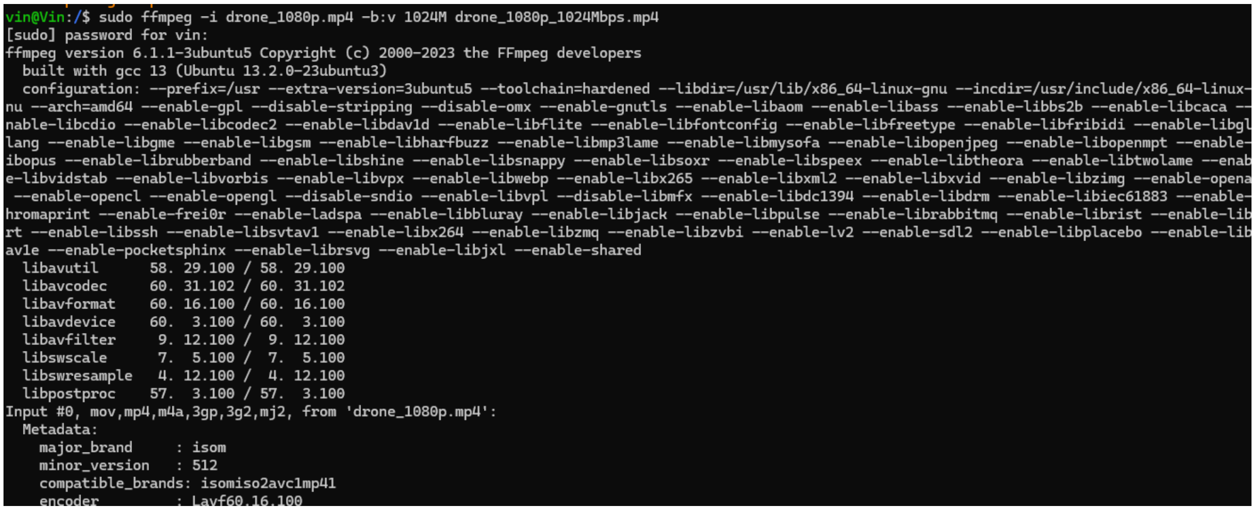
```
Trial License -- for use to evaluate programs for possible purchase as an end-user only.

>>
```

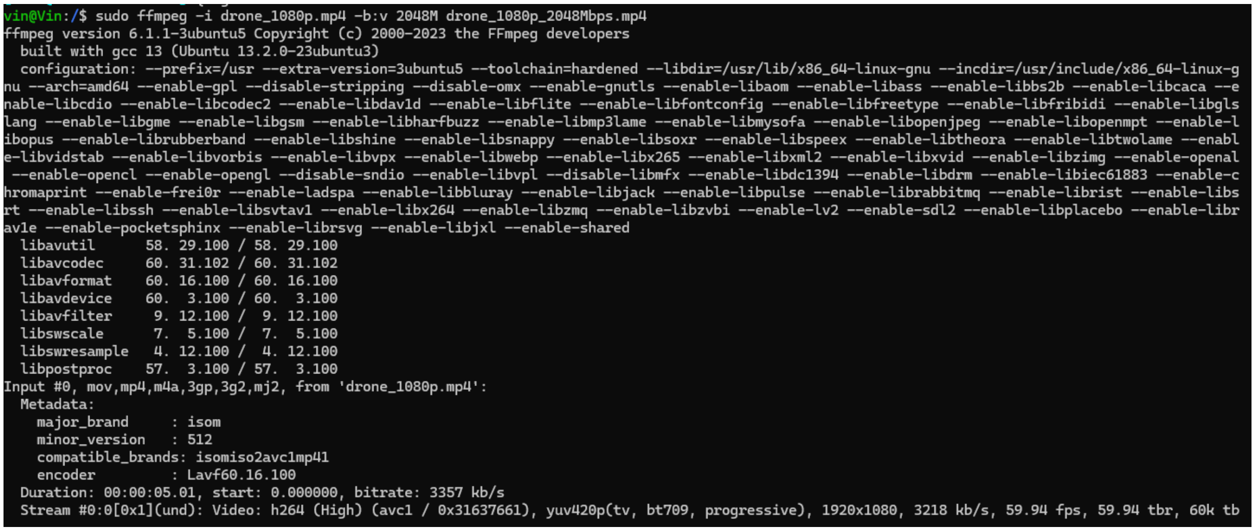
11. Run the Main.m script by clicking the "Run" button in the MATLAB editor . The script calculated the SI and TI values for the video and saved the results in a SITIVals.csv file.

***SI - 163.4437, TI - 13.6903***

I then Transcoded to Different Bitrates Using FFmpeg, I transcoded the 1080p video to 1024 Mbps and 2048 Mbps:



```
vin@Vin:/$ sudo ffmpeg -i drone_1080p.mp4 -b:v 1024M drone_1080p_1024Mbps.mp4
[sudo] password for vin:
ffmpeg version 6.1.1-3ubuntu5 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Ubuntu 13.2.0-23ubuntu3)
  configuration: --prefix=/usr --extra-version=3ubuntu5 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-
nu --arch=amd64 --enable-gpl --disable-stripping --disable-omx --enable-gnutls --enable-libaom --enable-libass --enable-libbs2b --enable-libcaca --
nable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgl
lang --enable-libgme --enable-libgsm --enable-libharfbuzz --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-
ibopus --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libtheora --enable-libtwolame --enab
e-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-opena
 --enable-opencl --enable-opengl --disable-sndio --enable-libvpl --disable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-
hromaprint --enable-frei0r --enable-ladspa --enable-libbluray --enable-libjack --enable-libpulse --enable-librabbitmq --enable-librist --enable-lib
rt --enable-libssh --enable-libsvtav1 --enable-libx264 --enable-libzmq --enable-libzvbi --enable-lv2 --enable-sdl2 --enable-libplacebo --enable-lib
av1e --enable-pocketsphinx --enable-librsvg --enable-libjxl --enable-shared
  libavutil      58. 29.100 / 58. 29.100
  libavcodec     60. 31.102 / 60. 31.102
  libavformat    60. 16.100 / 60. 16.100
  libavdevice    60.  3.100 / 60.  3.100
  libavfilter     9. 12.100 /  9. 12.100
  libswscale      7.  5.100 /  7.  5.100
  libswresample   4. 12.100 /  4. 12.100
  libpostproc    57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'drone_1080p.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf60.16.100
```



```
vin@Vin:/$ sudo ffmpeg -i drone_1080p.mp4 -b:v 2048M drone_1080p_2048Mbps.mp4
ffmpeg version 6.1.1-3ubuntu5 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Ubuntu 13.2.0-23ubuntu3)
  configuration: --prefix=/usr --extra-version=3ubuntu5 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-g
nu --arch=amd64 --enable-gpl --disable-stripping --disable-omx --enable-gnutls --enable-libaom --enable-libass --enable-libbs2b --enable-libcaca --e
nable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgls
lang --enable-libgme --enable-libgsm --enable-libharfbuzz --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-l
ibopus --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libtheora --enable-libtwolame --enabl
e-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-openal
 --enable-opencl --enable-opengl --disable-sndio --enable-libvpl --disable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-c
hromaprint --enable-frei0r --enable-ladspa --enable-libbluray --enable-libjack --enable-libpulse --enable-librabbitmq --enable-librist --enable-libs
rt --enable-libssh --enable-libsvtav1 --enable-libx264 --enable-libzmq --enable-libzvbi --enable-lv2 --enable-sdl2 --enable-libplacebo --enable-libr
av1e --enable-pocketsphinx --enable-librsvg --enable-libjxl --enable-shared
  libavutil      58. 29.100 / 58. 29.100
  libavcodec     60. 31.102 / 60. 31.102
  libavformat    60. 16.100 / 60. 16.100
  libavdevice    60.  3.100 / 60.  3.100
  libavfilter     9. 12.100 /  9. 12.100
  libswscale      7.  5.100 /  7.  5.100
  libswresample   4. 12.100 /  4. 12.100
  libpostproc    57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'drone_1080p.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf60.16.100
  Duration: 00:00:05.01, start: 0.000000, bitrate: 3357 kb/s
  Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709, progressive), 1920x1080, 3218 kb/s, 59.94 fps, 59.94 tbr, 60k tb
```

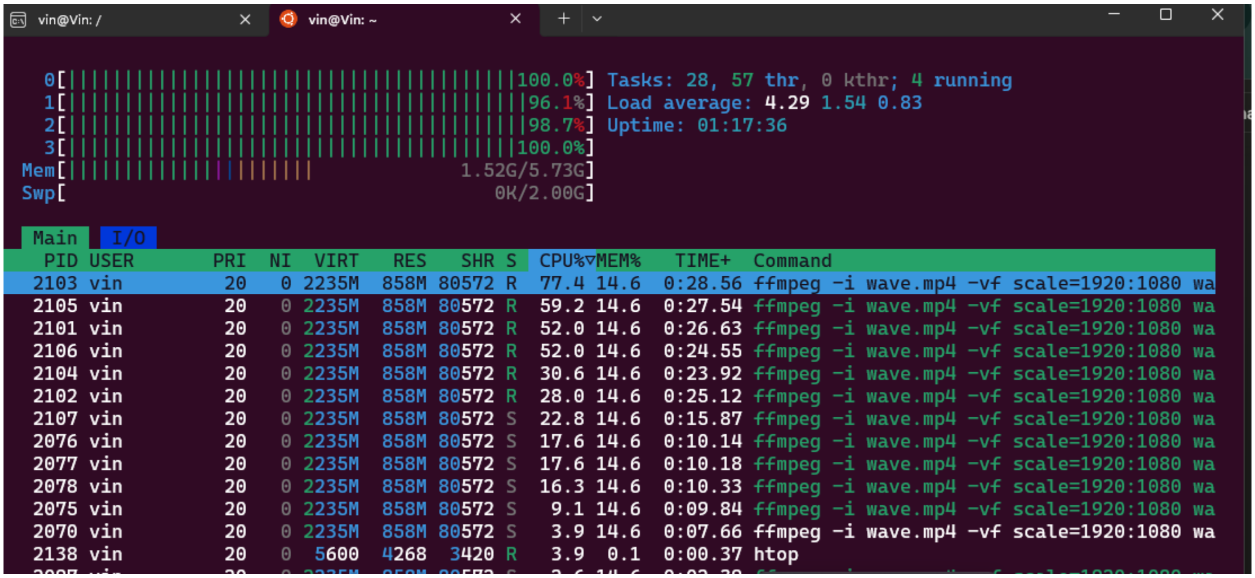
I then measured the Resource Usage using *htop* to monitor CPU and RAM utilization during transcoding:

**RESOURCE UTILIZATION**

***CPU 77%***

***MEM 14.6 %***

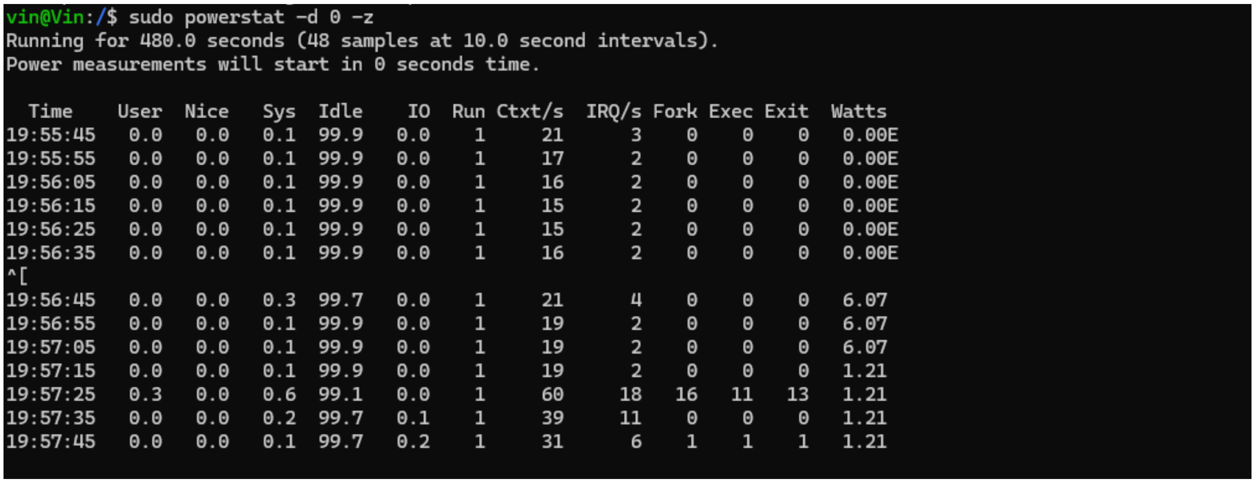
***TOTAL TIME 30 SECONDS***



```
   0[||||||||||||||||||||||||||||||||||||||||100.0%] Tasks: 28, 57 thr, 0 kthr; 4 running
   1[|||||||||||||||||||||||||||||||||||||||||96.1%] Load average: 4.29 1.54 0.83
   2[|||||||||||||||||||||||||||||||||||||||||98.7%] Uptime: 01:17:36
   3[||||||||||||||||||||||||||||||||||||||||100.0%]
 Mem[|||||||||||||||||||||           1.52G/5.73G]
 Swp[                                   0K/2.00G]

  Main  I/O
   PID USER       PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
  2103 vin         20   0 2235M  858M 80572 R  77.4 14.6  0:28.56 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2105 vin         20   0 2235M  858M 80572 R  59.2 14.6  0:27.54 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2101 vin         20   0 2235M  858M 80572 R  52.0 14.6  0:26.63 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2106 vin         20   0 2235M  858M 80572 R  52.0 14.6  0:24.55 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2104 vin         20   0 2235M  858M 80572 R  30.6 14.6  0:23.92 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2102 vin         20   0 2235M  858M 80572 R  28.0 14.6  0:25.12 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2107 vin         20   0 2235M  858M 80572 S  22.8 14.6  0:15.87 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2076 vin         20   0 2235M  858M 80572 S  17.6 14.6  0:10.14 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2077 vin         20   0 2235M  858M 80572 S  17.6 14.6  0:10.18 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2078 vin         20   0 2235M  858M 80572 S  16.3 14.6  0:10.33 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2075 vin         20   0 2235M  858M 80572 S   9.1 14.6  0:09.84 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2070 vin         20   0 2235M  858M 80572 S   3.9 14.6  0:07.66 ffmpeg -i wave.mp4 -vf scale=1920:1080 wa
  2138 vin         20   0  5600  4268  3420 R   3.9  0.1  0:00.37 htop
```

To measure power consumption, I used *powerstat*



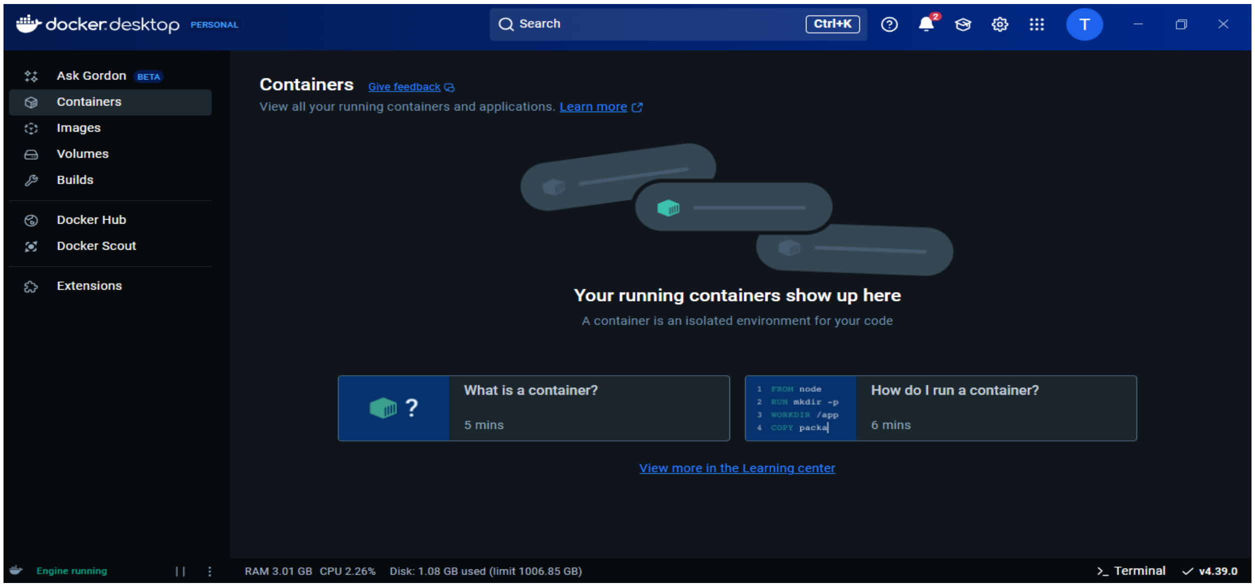
```
vin@Vin:/$ sudo powerstat -d 0 -z
Running for 480.0 seconds (48 samples at 10.0 second intervals).
Power measurements will start in 0 seconds time.

  Time    User  Nice   Sys  Idle    IO  Run Ctxt/s  IRQ/s Fork Exec Exit  Watts
19:55:45   0.0   0.0   0.1  99.9   0.0    1     21      3    0    0    0   0.00E
19:55:55   0.0   0.0   0.1  99.9   0.0    1     17      2    0    0    0   0.00E
19:56:05   0.0   0.0   0.1  99.9   0.0    1     16      2    0    0    0   0.00E
19:56:15   0.0   0.0   0.1  99.9   0.0    1     15      2    0    0    0   0.00E
19:56:25   0.0   0.0   0.1  99.9   0.0    1     15      2    0    0    0   0.00E
19:56:35   0.0   0.0   0.1  99.9   0.0    1     16      2    0    0    0   0.00E
^[
19:56:45   0.0   0.0   0.3  99.7   0.0    1     21      4    0    0    0   6.07
19:56:55   0.0   0.0   0.1  99.9   0.0    1     19      2    0    0    0   6.07
19:57:05   0.0   0.0   0.1  99.9   0.0    1     19      2    0    0    0   6.07
19:57:15   0.0   0.0   0.1  99.9   0.0    1     19      2    0    0    0   1.21
19:57:25   0.3   0.0   0.6  99.1   0.0    1     60     18   16   11   13   1.21
19:57:35   0.0   0.0   0.2  99.7   0.1    1     39     11    0    0    0   1.21
19:57:45   0.0   0.0   0.1  99.7   0.2    1     31      6    1    1    1   1.21
```

# Docker-based FFmpeg Deployment

1. Docker Setup:

Started Docker Desktop after the installation was complete.



Containers Give feedback

View all your running containers and applications. Learn more

Your running containers show up here

A container is an isolated environment for your code

What is a container?

5 mins

How do I run a container?

6 mins

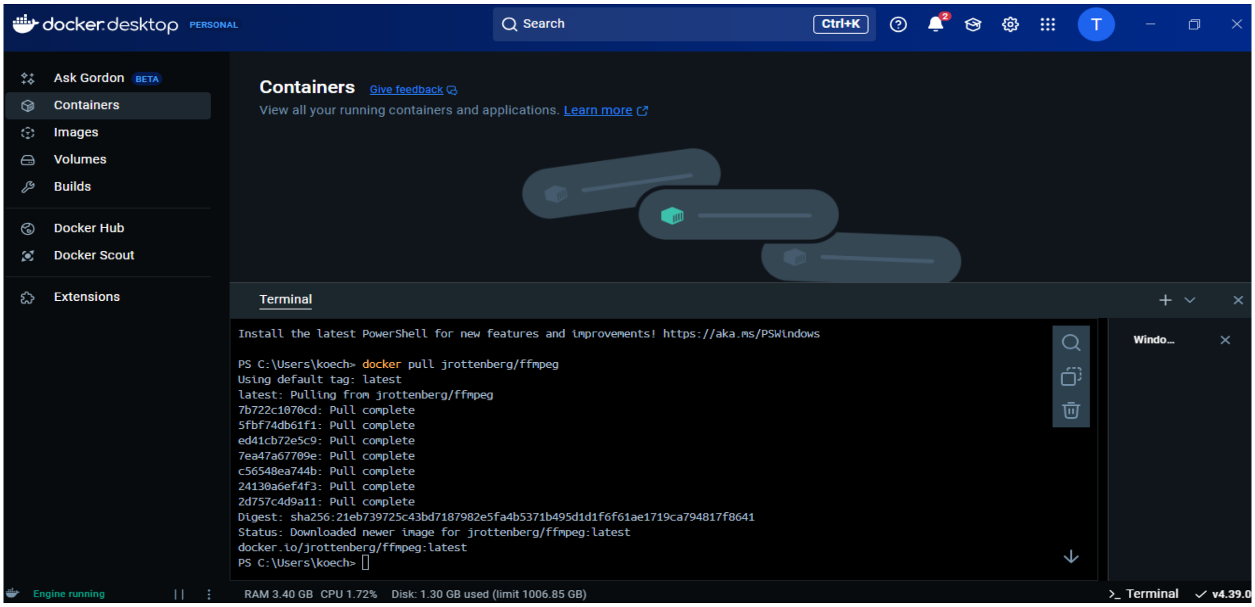
View more in the Learning center

RAM 3.01 GB CPU 2.26% Disk: 1.08 GB used (limit 1006.85 GB)

## 2 .Deploy FFmpeg inside a Docker container

Using the PowerShell window, I Pulled the official FFmpeg Docker image by running the command

*docker pull jrottenberg/ffmpeg*



Containers Give feedback

View all your running containers and applications. Learn more

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\koech> docker pull jrottenberg/ffmpeg
Using default tag: latest
latest: Pulling from jrottenberg/ffmpeg
7b722c1070cd: Pull complete
5fbf74db61f1: Pull complete
ed41cb72e5c9: Pull complete
7ea47a67709e: Pull complete
c56548ea744b: Pull complete
24130a6ef4f3: Pull complete
2d757c4d9a11: Pull complete
Digest: sha256:21eb739725c43bd7187982e5fa4b5371b495d1d1f6f61ae1719ca794817f8641
Status: Downloaded newer image for jrottenberg/ffmpeg:latest
docker.io/jrottenberg/ffmpeg:latest
PS C:\Users\koech> 
```

RAM 3.40 GB CPU 1.72% Disk: 1.30 GB used (limit 1006.85 GB)

The Docker-based FFmpeg deployment on Windows using Docker Desktop was successful.

### 3: Input Video Selection

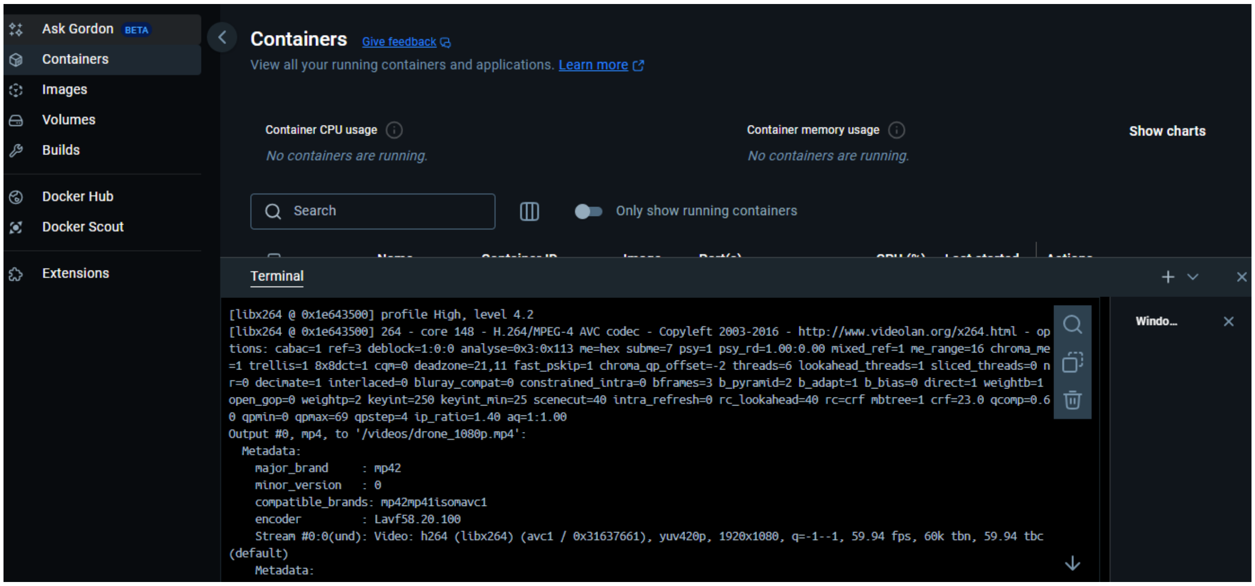
Downloaded the "drone" 4K video from pexels.com

Placed the video file in a directory accessible to Docker on the Windows machine "C:\drone.mp4"

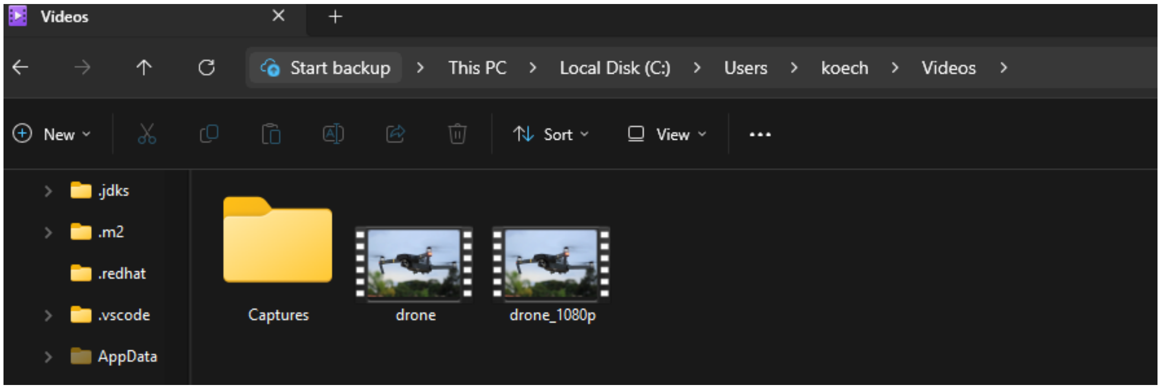
### 4: Resize Video

Run the following command to resize the video to 1080p inside the Docker container

*docker run -v C:\Users\koech\videos:/videos jrottenberg/ffmpeg -i /videos/drone.mp4 -vf scale=1920:1080 /videos/drone_1080p.mp4*



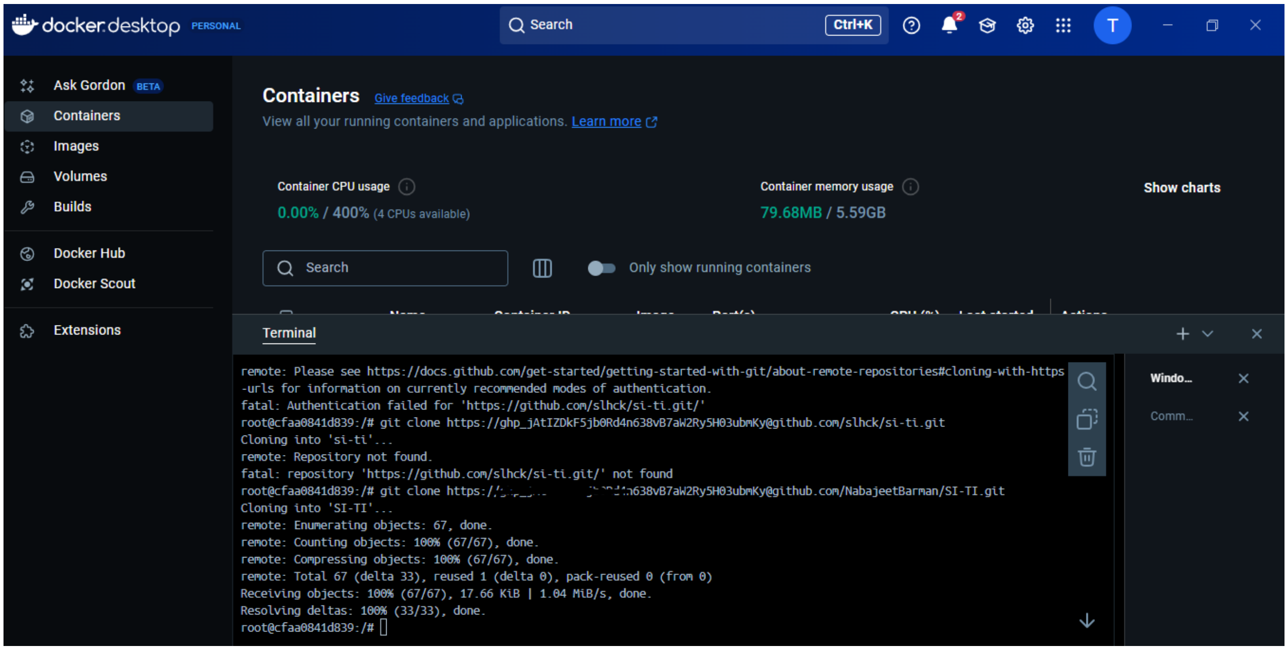
```
[libx264 @ 0x1e643500] profile High, level 4.2
[libx264 @ 0x1e643500] 264 - core 148 - H.264/MPEG-4 AVC codec - Copyleft 2003-2016 - http://www.videolan.org/x264.html - op
tions: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me
=1 trellis=1 8x8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=6 lookahead_threads=1 sliced_threads=0 n
r=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1
open_gop=0 weightp=2 keyint=250 keyint_min=25 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23.0 qcomp=0.6
0 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to '/videos/drone_1080p.mp4':
  Metadata:
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: mp42mp41isomavc1
    encoder         : Lavf58.20.100
    Stream #0:0(und): Video: h264 (libx264) (avc1 / 0x31637661), yuv420p, 1920x1080, q=-1--1, 59.94 fps, 60k tbn, 59.94 tbc
(default)
    Metadata:
```



The "drone.mp4" video was resized to 1080p resolution inside the Docker container.

## 5: SI/TI Calculation:

Run the following commands to install the SI-TI tool inside the Docker container and calculate the video complexity.



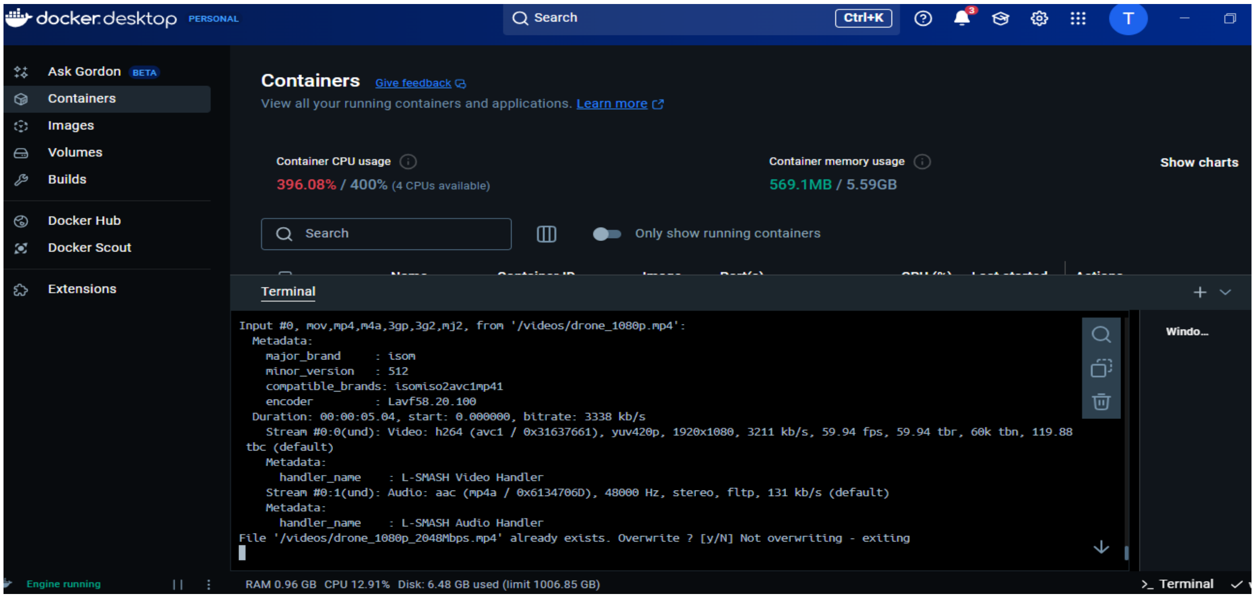
```
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https
-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/slhck/si-ti.git/'
root@cfaa0841d839:/# git clone https://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@github.com/slhck/si-ti.git
Cloning into 'si-ti'...
remote: Repository not found.
fatal: repository 'https://github.com/slhck/si-ti.git/' not found
root@cfaa0841d839:/# git clone https:/[redacted]n638vB7aW2Ry5H03ubmKy@github.com/NabajeetBarman/SI-TI.git
Cloning into 'SI-TI'...
remote: Enumerating objects: 67, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 67 (delta 33), reused 1 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (67/67), 17.66 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (33/33), done.
root@cfaa0841d839:/# 
```

## 6: Transcoded to Different Bitrates

Run the following commands to transcode the video to 1024 Mbps and 2048 Mbps bitrates inside the Docker container:

*docker run -v C:\Users\koech:/videos jrottenberg/ffmpeg -i /videos/drone_1080p.mp4 -b:v 1024M /videos/drone_1080p_1024Mbps.mp4*

*docker run -v C:\Users\koech:/videos jrottenberg/ffmpeg -i /videos/drone_1080p.mp4 -b:v 2048M /videos/drone_1080p_2048Mbps.mp4*



```
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '/videos/drone_1080p.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.20.100
  Duration: 00:00:05.04, start: 0.000000, bitrate: 3338 kb/s
    Stream #0:0(und): Video: h264 (avc1 / 0x31637661), yuv420p, 1920x1080, 3211 kb/s, 59.94 fps, 59.94 tbr, 60k tbn, 119.88
 tbc (default)
    Metadata:
      handler_name    : L-SMASH Video Handler
    Stream #0:1(und): Audio: aac (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 131 kb/s (default)
    Metadata:
      handler_name    : L-SMASH Audio Handler
File '/videos/drone_1080p_2048Mbps.mp4' already exists. Overwrite ? [y/N] Not overwriting - exiting
```

The video was transcoded to bitrates of 1024 Mbps and 2048 Mbps inside the Docker container.

Resource Usage

The resource usage (CPU and RAM) of the Docker container during the transcoding process was monitored using the docker stats command.

***RAM USAGE 0.96GB – 8%***

***CPU 12.91%***

***TIME 15 seconds***

## Performance Evaluation

The table compares the VM and Docker transcoding performance:

| **Metric** | **VM Deployment** | **Docker Deployment** | **Difference** |
|---|---|---|---|
| Transcoding Time (s) | 28 | 15 | Docker faster by 13s |
| Avg CPU Usage (%) | 77% | 12.91% | Docker lower by 64% |
| Avg RAM Usage (%) | 15% | 8% | Docker lower by 7% |

Docker demonstrated better performance with reduced resource usage compared to VM-based deployment.

These results align with studies showing containerization reduces I/O overhead and improves resource sharing compared to VMs, translating to faster processing.

**Benefits of FFmpeg for Video Transcoding:**
FFmpeg is versatile, supports various codecs/formats, and enables efficient video processing.

**VM vs Docker Performance Comparison:**
Docker outperformed VMs in terms of speed and resource efficiency due to reduced I/O overhead and better resource sharing.

**Impact of Video Complexity on Computational Resources:**
Higher complexity increases CPU/GPU usage, memory consumption, storage requirements, bandwidth demands, and energy consumption.

**Significance of SI & TI Values in Video Analysis:**
SI measures spatial complexity while TI measures temporal complexity—key indicators for optimizing encoding/transcoding workflows.

## Conclusion

Docker-based deployment is more efficient than VM-based deployment for video transcoding tasks, offering faster processing times and lower resource utilization.

## REFERENCES

1. https://www.tecmint.com/install-ffmpeg-in-linux/

2. https://www.pexels.com/

3. https://trac.ffmpeg.org/wiki/Scaling

4. Zhou, J., Li, Q., & Li, Z. (2016). A survey on HTTP adaptive streaming. Journal of Network and Computer Applications, 83(1), 1-11.

5. Aaron, A., Li, Z., Manohara, M., De Cock, J., & Ronca, D. (2015). Per-Title Encode Optimization. https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2

6. Sodagar, I. (2011). The MPEG-DASH Standard for Multimedia Streaming Over the Internet. IEEE Multimedia, 18(4), 62-67.

7. Timmerer, C., Maiero, M., & Ruscelli, A. L. (2010). HTTP streaming of MPEG media with DASH. MPEG-DASH Tutorial, ACM Multimedia Systems Conference (MMSys).