WELCOME

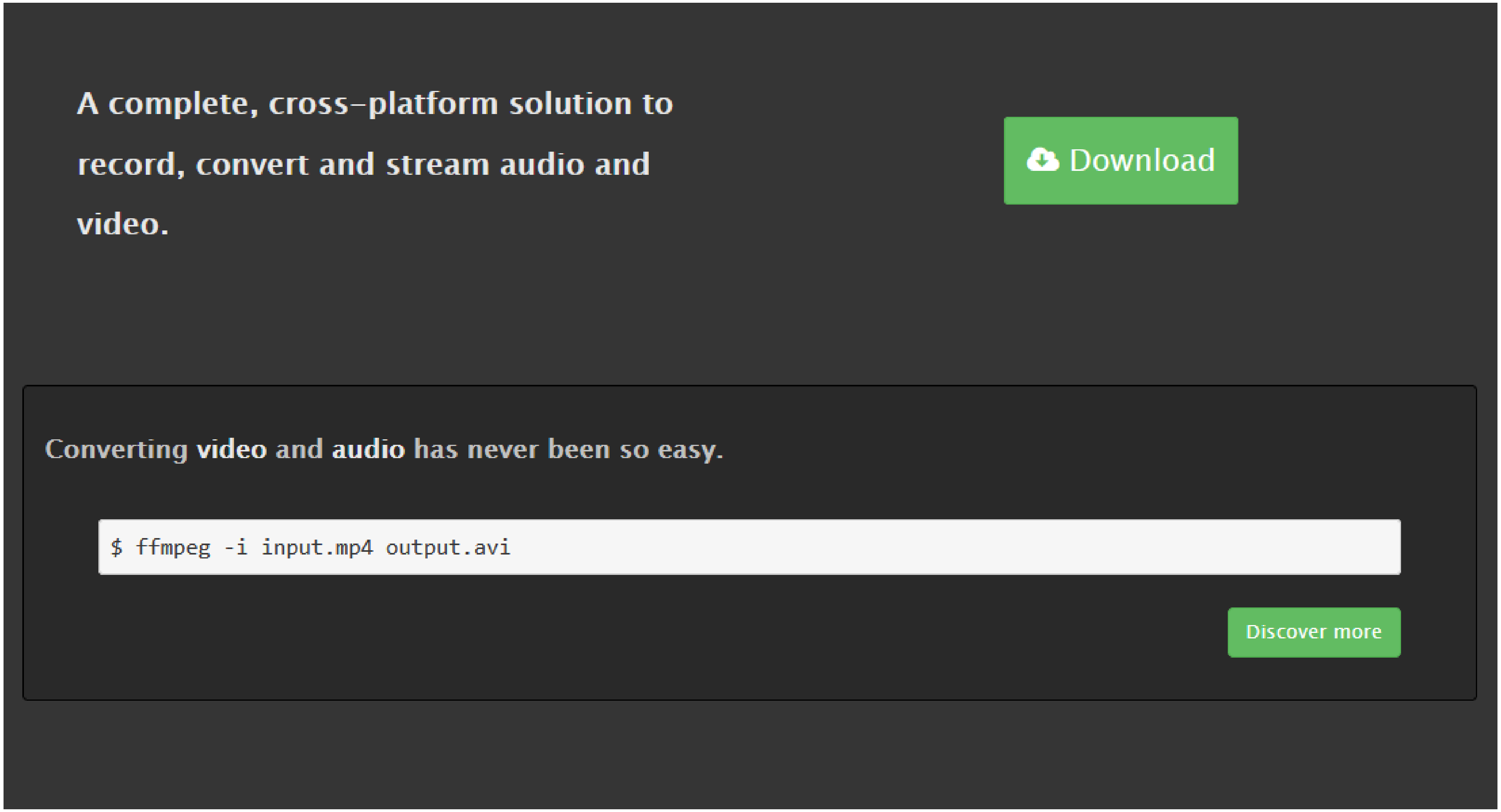
FFmpeg

Table of contents

- What is FFmpeg
- What can it do?
- Example in Command prompt
- FFmpy

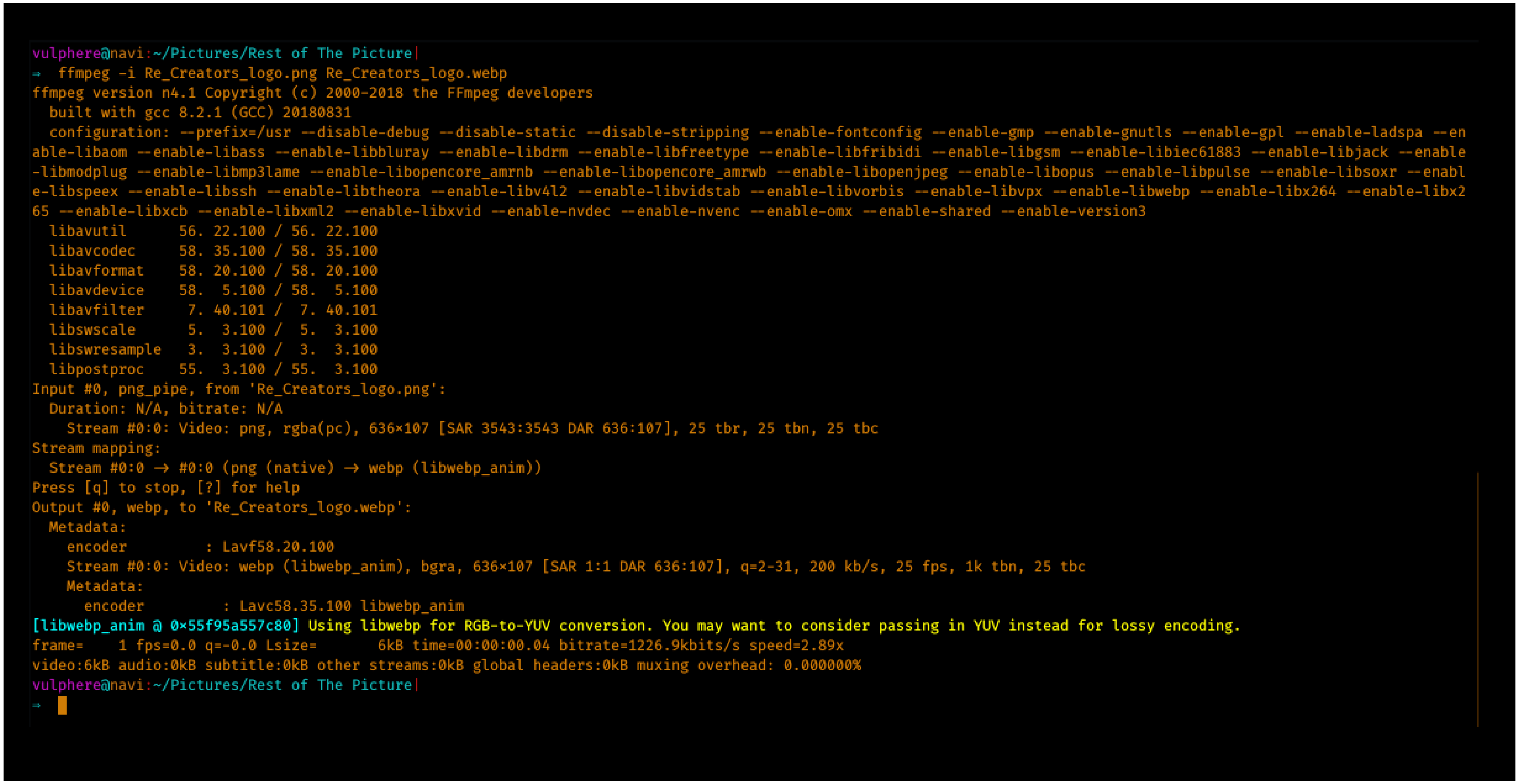
What is FFmpeg

FFmpeg.org

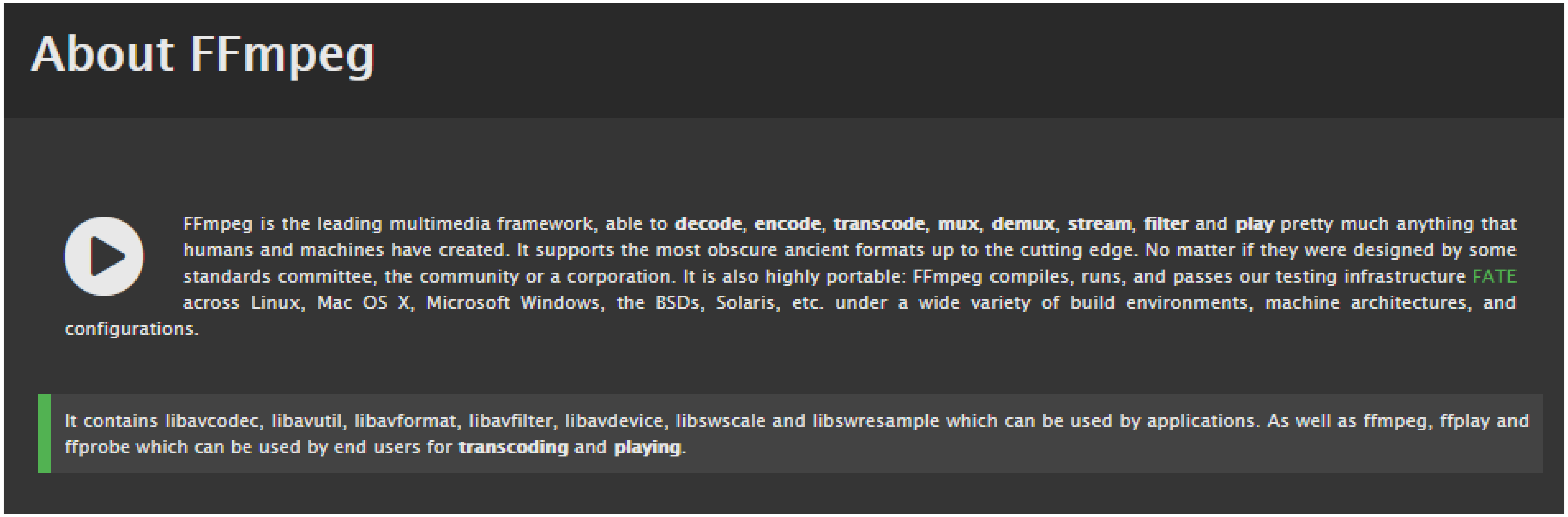

A complete, cross-platform solution to record, convert and stream audio and video.

Converting **video** and **audio** has never been so easy.

```
$ ffmpeg -i input.mp4 output.avi
```



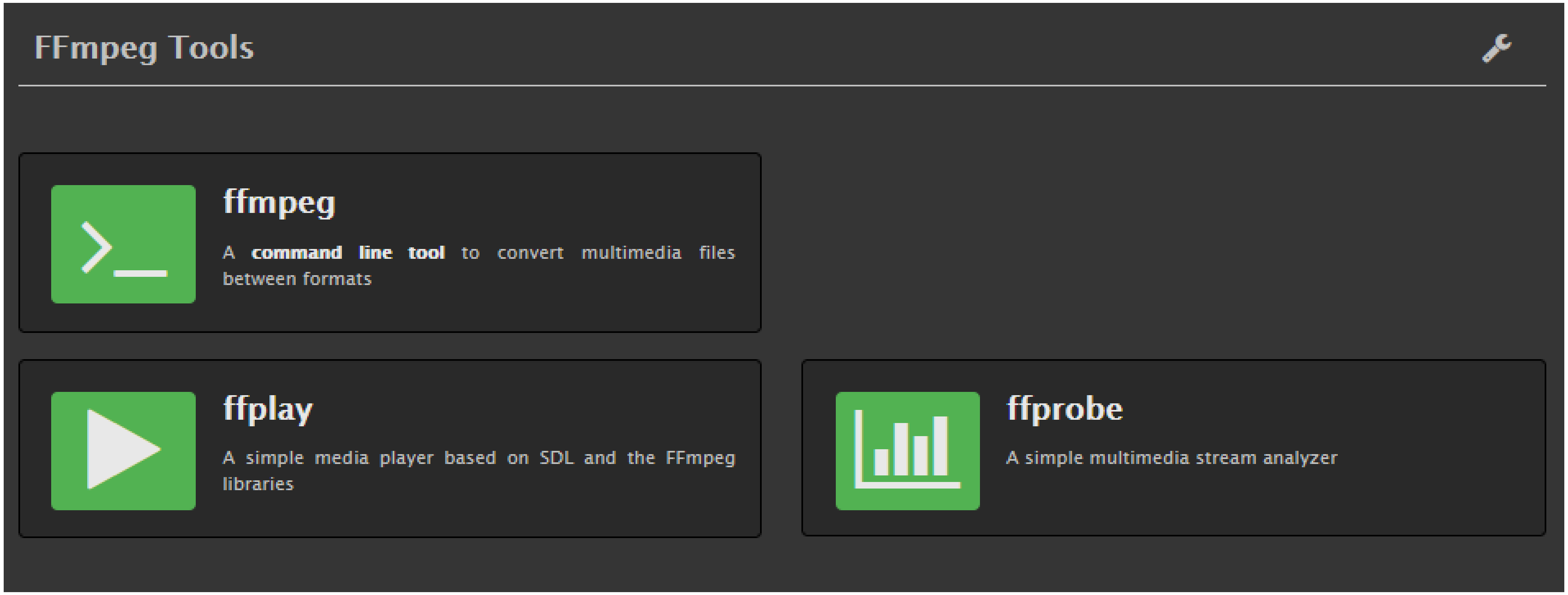
```
vulphere@navi:~/Pictures/Rest of The Picture|
⇒  ffmpeg -i Re_Creators_logo.png Re_Creators_logo.webp
ffmpeg version n4.1 Copyright (c) 2000-2018 the FFmpeg developers
  built with gcc 8.2.1 (GCC) 20180831
  configuration: --prefix=/usr --disable-debug --disable-static --disable-stripping --enable-fontconfig --enable-gmp --enable-gnutls --enable-gpl --enable-ladspa --en
able-libaom --enable-libass --enable-libbluray --enable-libdrm --enable-libfreetype --enable-libfribidi --enable-libgsm --enable-libiec61883 --enable-libjack --enable
-libmodplug --enable-libmp3lame --enable-libopencore_amrnb --enable-libopencore_amrwb --enable-libopenjpeg --enable-libopus --enable-libpulse --enable-libsoxr --enabl
e-libspeex --enable-libssh --enable-libtheora --enable-libv4l2 --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx264 --enable-libx2
65 --enable-libxcb --enable-libxml2 --enable-libxvid --enable-nvdec --enable-nvenc --enable-omx --enable-shared --enable-version3
  libavutil      56. 22.100 / 56. 22.100
  libavcodec     58. 35.100 / 58. 35.100
  libavformat    58. 20.100 / 58. 20.100
  libavdevice    58.  5.100 / 58.  5.100
  libavfilter     7. 40.101 /  7. 40.101
  libswscale      5.  3.100 /  5.  3.100
  libswresample   3.  3.100 /  3.  3.100
  libpostproc    55.  3.100 / 55.  3.100
Input #0, png_pipe, from 'Re_Creators_logo.png':
  Duration: N/A, bitrate: N/A
    Stream #0:0: Video: png, rgba(pc), 636x107 [SAR 3543:3543 DAR 636:107], 25 tbr, 25 tbn, 25 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (png (native) -> webp (libwebp_anim))
Press [q] to stop, [?] for help
Output #0, webp, to 'Re_Creators_logo.webp':
  Metadata:
    encoder         : Lavf58.20.100
    Stream #0:0: Video: webp (libwebp_anim), bgra, 636x107 [SAR 1:1 DAR 636:107], q=2-31, 200 kb/s, 25 fps, 1k tbn, 25 tbc
    Metadata:
      encoder         : Lavc58.35.100 libwebp_anim
[libwebp_anim @ 0x55f95a557c80] Using libwebp for RGB-to-YUV conversion. You may want to consider passing in YUV instead for lossy encoding.
frame=    1 fps=0.0 q=-0.0 Lsize=       6kB time=00:00:00.04 bitrate=1226.9kbits/s speed=2.89x
video:6kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.000000%
vulphere@navi:~/Pictures/Rest of The Picture|
⇒
```



About FFmpeg

FFmpeg is the leading multimedia framework, able to **decode**, **encode**, **transcode**, **mux**, **demux**, **stream**, **filter** and **play** pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.

It contains libavcodec, libavutil, libavformat, libavfilter, libavdevice, libswscale and libswresample which can be used by applications. As well as ffmpeg, ffplay and ffprobe which can be used by end users for **transcoding** and **playing**.



FFmpeg Tools

ffmpeg

A **command line tool** to convert multimedia files between formats

ffplay

A simple media player based on SDL and the FFmpeg libraries

ffprobe

A simple multimedia stream analyzer

Video and audio editor

- CLI Command line interface
- Free – Open source
- FFmpeg
- FFplay – media player
- FFprobe – media information
- Backbone

SYNTAX

```
ffmpeg -i <path to inputfile> -vf scale=1920:1080 -t 00:00:00 -ss
00:01:00 –pix_fmt yuv420p -an <path to output.container>
```

Lossless H.264

You can use `-crf 0` to create a lossless video. Two useful presets for this are `ultrafast` or `veryslow` since either a fast encoding speed or best c

Fast encoding example:

```
ffmpeg -i input -c:v libx264 -preset ultrafast -crf 0 output.mkv
```

Best compression example:

```
ffmpeg -i input -c:v libx264 -preset veryslow -crf 0 output.mkv
```

Note that lossless output files will likely be huge, and most non-FFmpeg based players will not be able to decode lossless. Therefore, if compatibility or f

Tip: If you're looking for an output that is roughly "visually lossless" but not technically lossless, use a `-crf` value of around 17 or 18 (you'll have
indistinguishable from the source and not result in a huge, possibly incompatible file like true lossless mode.

Overwriting default preset settings

While `-preset` chooses the best possible settings for you, you can overwrite these with the `x264-params` option, or by using the libx264 private opt
you know what you are doing. The presets were created by the x264 developers and tweaking values to get a better output is usually a waste of time.

Example:

```
ffmpeg -i input -c:v libx264 -preset slow -crf 22 -x264-params keyint=123:min-keyint=20 -c:a copy output.mkv
```

Warning: Do not use the option `x264opts`, as it will eventually be removed. Use `x264-params` instead.

Additional Information & Tips

CBR (Constant Bit Rate)

There is no native or true CBR mode, but you can "simulate" a constant bit rate setting by tuning the parameters of a one-pass average bitrate encode

WHAT CAN('T) IT DO

- (re)coding & muxing:
- Scaling / cropping
- Cutting / editing
- Compression
- Colorspace / chroma subsampling / keying
- Framerates
- Text / subtitling
- Image formats (Jpeg, Tiff, PNG, ...)

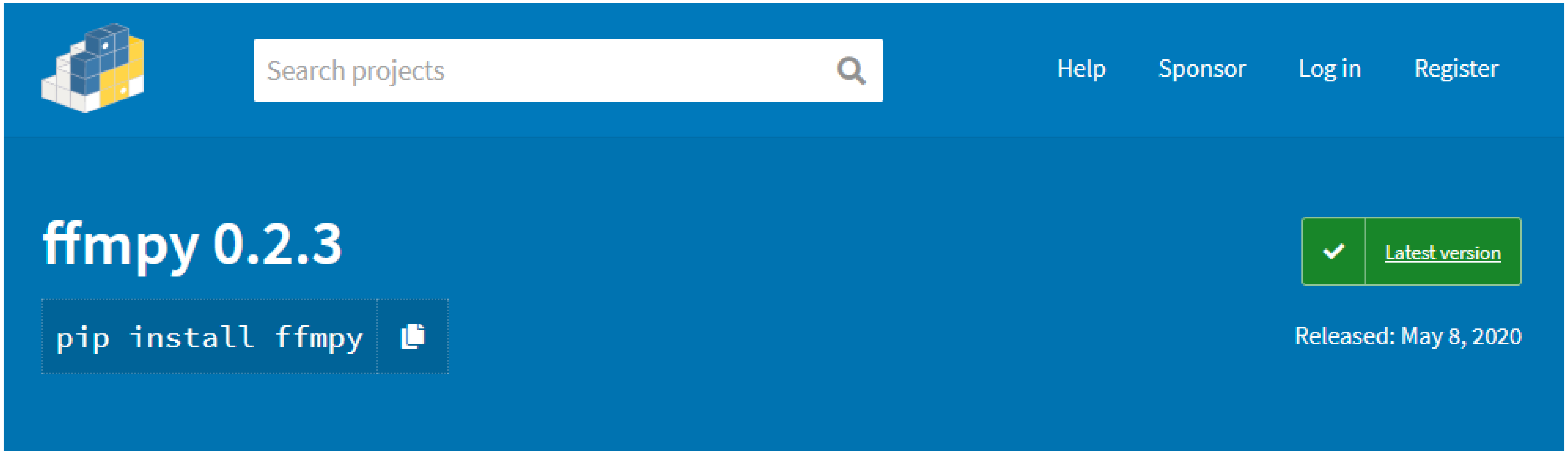
WHAT CAN('T) IT DO

- Normalisation (loudness) / Filtering
- De-interlacing / Denoising
- OCR
- Visualisations (vectorscope, waveforms, color histogram)
- Black & white / negative
- ...

Example

FFmpy

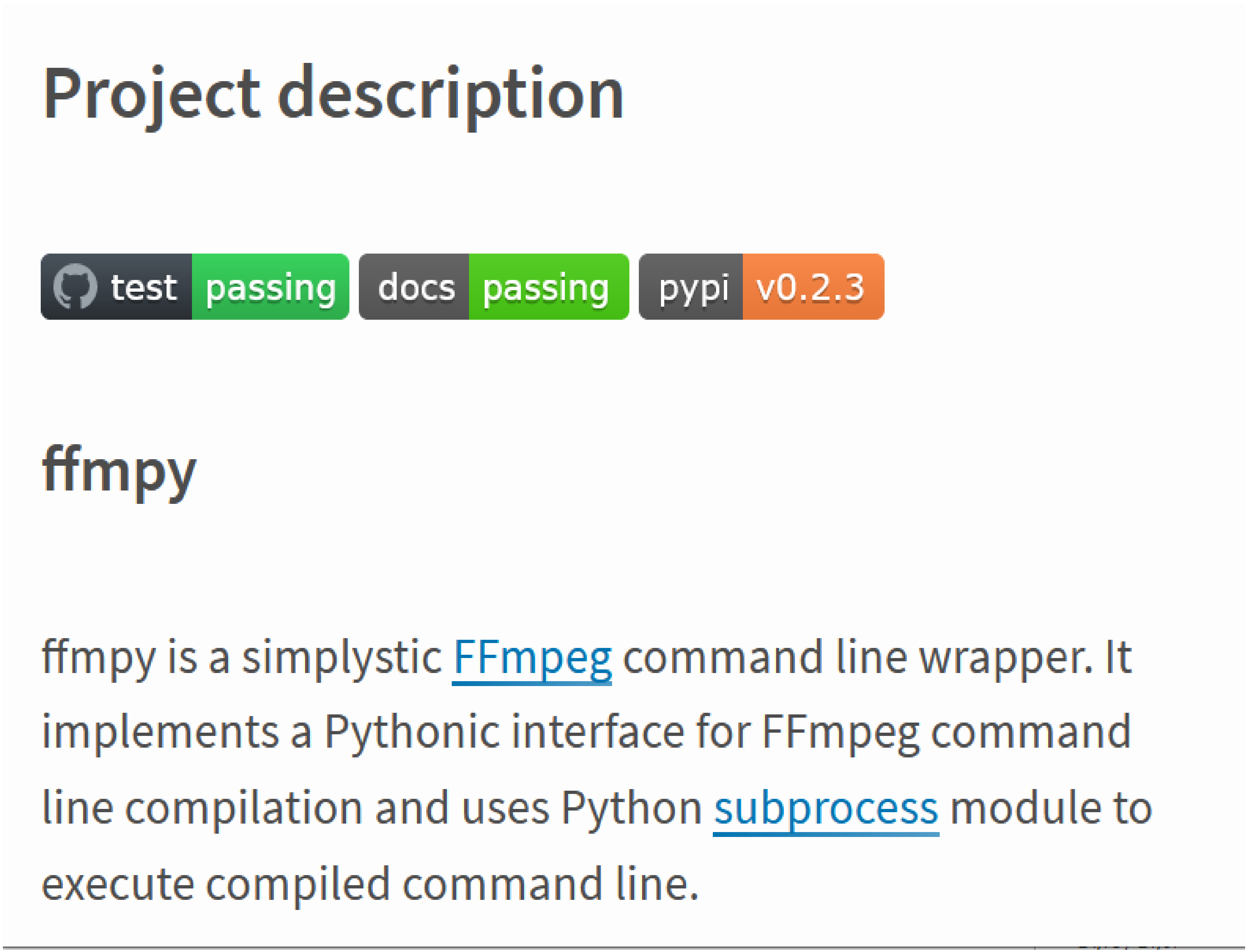
https://pypi.org/project/ffmpy/



ffmpy 0.2.3

```
pip install ffmpy
```

Released: May 8, 2020



Project description

test passing docs passing pypi v0.2.3

ffmpy

ffmpy is a simplystic FFmpeg command line wrapper. It implements a Pythonic interface for FFmpeg command line compilation and uses Python subprocess module to execute compiled command line.

Installation

You guessed it:

```
pip install ffmpy
```

Quick example

```
>>> import ffmpy
>>> ff = ffmpy.FFmpeg(
...     inputs={'input.mp4': None},
...     outputs={'output.avi': None}
... )
>>> ff.run()
```

This will take `input.mp4` file in the current directory as the input, change the video container from MP4 to AVI without changing any other video parameters and create a new output file `output.avi` in the current directory.

FFthanks!