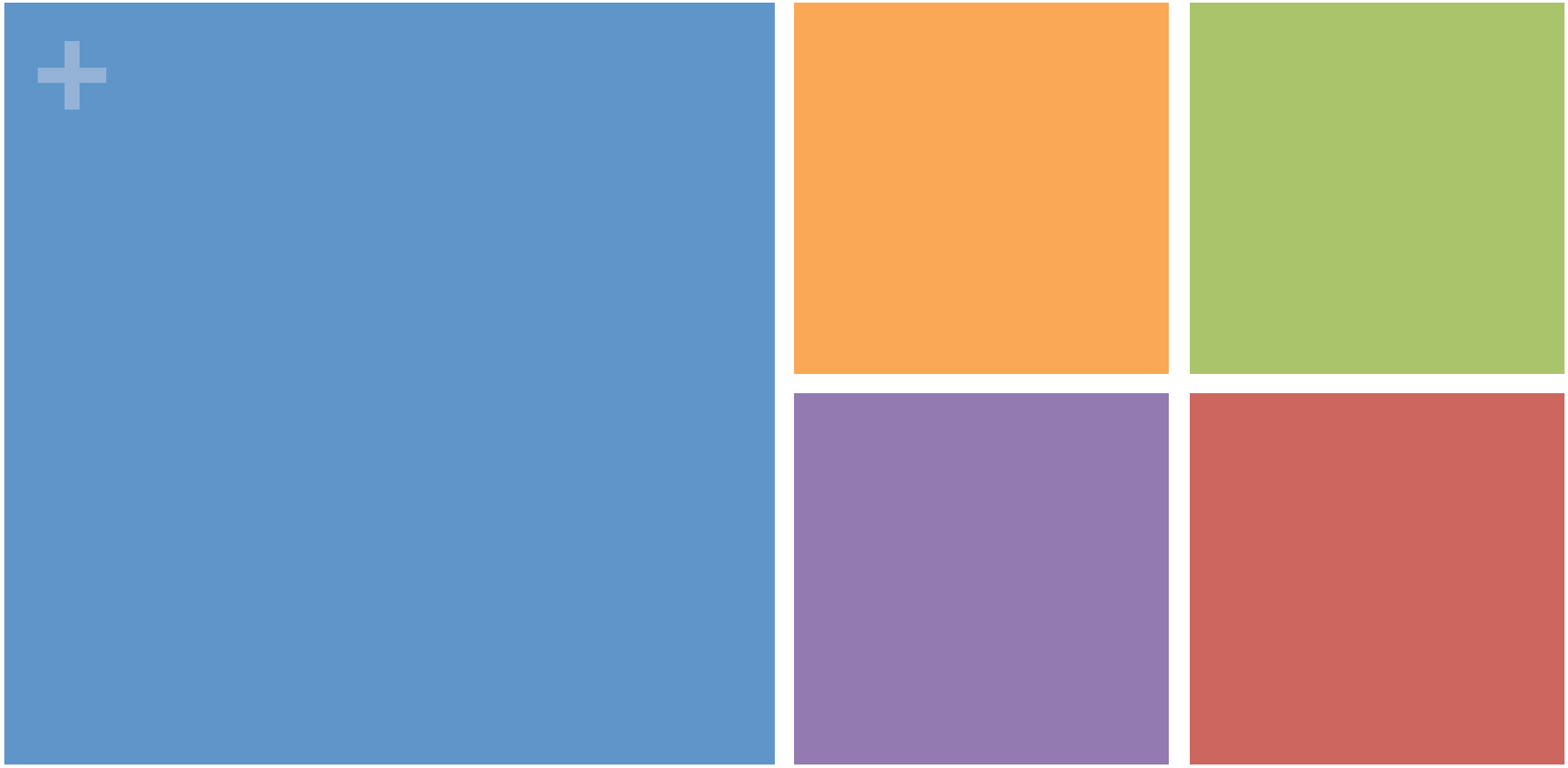


FFMPEG on the IBM Cloud

Team 8: Rishi Ishairzay, Puloma De, Andrew Hwang

Content

- Intro to FFMPEG
- Basics of Video
- The Server
- Basics of FFMPEG
- Little more in-depth with FFMPEG
- Examples

FFMPEG

- Stands for *Fast* Forward MPEG
- Completely open source
- Used by thousands of video-enabled websites
- Supports encoding/modifying video/audio/image formats
- Extremely thorough set of options

Audio Encoding

- Audio Sample Rate
  - Number of samples per second (e.g. 48 kHz)
- Audio Bitrate
  - How much data is used to represent a single second of audio (e.g. 128kbps)
- Audio Codec
  - Specific algorithm to use for representing the audio (e.g. MP3, FLAC, etc.)

Video Encoding

- Frame Rate
  - Amount of frames per second
- Video Dimensions
  - Height and width of each frame in the video
- Cropping/Padding
  - Amount of pixels to chop off from the edge of the video or amount of pixels to add to the edge of the video.
- Video Codec
  - Specific algorithm used to represent the data

How we did it

Getting FFMPEG installed in 5 steps

- curl http://ffmpeg.org/releases/ffmpeg-0.10.2.tar.gz | tar xz
- cd ffmpeg-0.10.2/
- ./configure --disable-yasm
- make
- make install

These steps (and more) have already been done for you on the server.

The standard FFMPEG install provides support for basic video and audio formats. Other codecs are supported through separate installations.

Working with FFMPEG

Basic Commands

- ffmpeg –i <Input> -s <Dimensions> …
  - Input – Source video file
  - Dimensions – Width x Height (e.g. 800x600)
- … –b:v <Bitrate> -codec:v <Codec> …
  - Bitrate – Video bitrate in kbits/s (e.g. 768kbps)
  - Codec – Video codec (e.g. libxvid)
- … –bt <Tolerance> -r <Frame rate> …
  - Tolerance – The acceptable variance in bitrate
  - Frame rate – The frame rate for the output file
- … -codec:a <Codec> -b:a <Bitrate>
  - Codec – Audio codec (e.g. mp3)
  - Bitrate – Audio bitrate in kbits/s

Let's see it in action

- ffmpeg -i input.mpeg -s 640x266-b:v 512k -bt 240k -codec:v libx264 -codec:a ac3 -b:a 192k output.avi
  - Takes in input.mpeg
  - Sets the size of the output video to 640x266px
  - Sets the bit rate of the video to 512kbps
  - Allows a fluctuation of 240kbps in video quality, which allows the quality to range from 392kbps (for low action scenes) to 632kbps (for high action scenes)
  - Sets the video codec to libx264
  - Sets the audio codec to ac3
  - Sets the audio bitrate to 192kbps
  - Outputs file to output.avi

Do it!

- Before running any commands, switch into your team directory ~/ffmpeg/teams/**<team name>**
  - If the directory doesn't exist, then make it!
- Try executing the command with your own input file (or use the one provided – ~/ffmpeg/transform.mov)
  - If you don't have a sample file to work with, you can download a movie trailer with the command below.
    - curl -A QuickTime -O http://trailers.apple.com/movies/paramount/transformers3/transformers3-sbspot_h720p.mov
- Should look something like this:

```
idcuser:~/ffmpeg/teams/team1$ ffmpeg -i ../../transform.mov
 -s 640x266 -b:v 512k -bt 240k -codec:v libx264 -codec:a ac
3 -b:a 192k output.avi
```

Congratulations!

You just started a video conversion with FFMPEG!

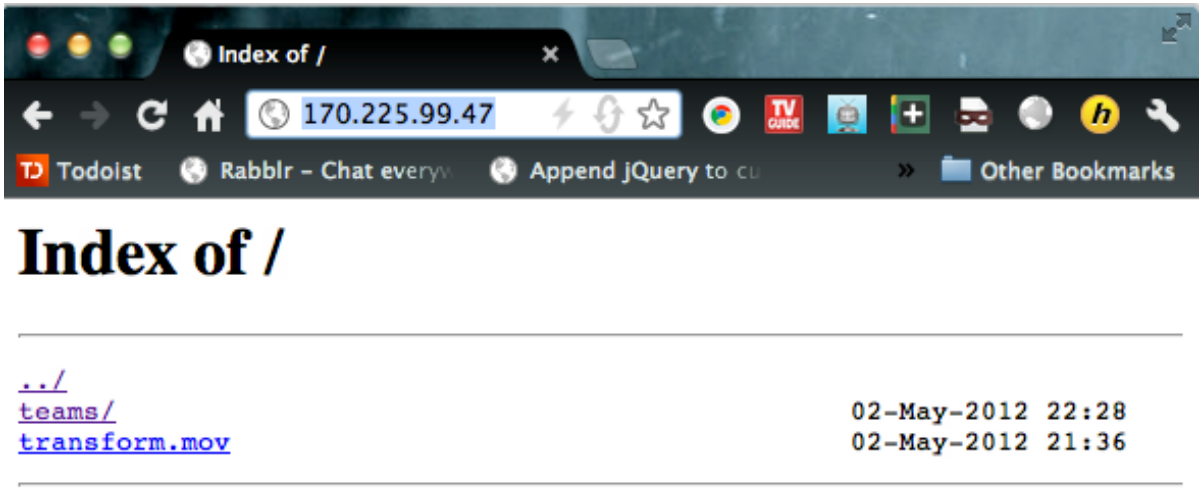
You should be seeing a screen like the one on the right, outlining how much progress has been made along with the metadata associated with the input and output streams.

P.S. The video you just created is the exact format that iPhones and iPads are capable of playing.

```
w:1280 h:532 pixfmt:yuv420p tb:1/1000000 sar:0/1 sws_param:
[scale @ 0x1574520] w:1280 h:532 fmt:yuv420p -> w:640 h:266 fmt:yuv420p flags:0x4
Incompatible sample format 's16' for codec 'ac3', auto-selecting format 'flt'
[libx264 @ 0x1591a80] using cpu capabilities: none!
[libx264 @ 0x1591a80] profile High, level 2.1
Output #0, avi, to 'output.avi':
  Metadata:
    major_brand     : qt
    minor_version   : 537199360
    compatible_brands: qt
    creation_time   : 2011-02-04 21:14:20
    ICMT            : Encoded and delivered by apple.com/trailers/
    comment-eng     : Encoded and delivered by apple.com/trailers/
    ICOP            : © 2011 Paramount Pictures. All Rights Reserved
    copyright-eng   : © 2011 Paramount Pictures. All Rights Reserved
    INAM            : Transformers
    title-eng       : Transformers
    ISFT            : Lavf53.32.100
    Stream #0:0(eng): Video: h264 (H264 / 0x34363248), yuv420p, 640x266, q=-1--1, 512 kb/s, 23.98 tbn,
 23.98 tbc
    Metadata:
      creation_time   : 2011-02-04 21:14:20
      handler_name    : ?Apple Alias Data Handler
    Stream #0:1(eng): Audio: ac3 ([0] [0][0] / 0x2000), 48000 Hz, stereo, flt, 192 kb/s
    Metadata:
      creation_time   : 2011-02-04 21:14:20
      handler_name    : ?Apple Alias Data Handler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 -> libx264)
  Stream #0:1 -> #0:1 (aac -> ac3)
Press [q] to stop, [?] for help
[h264 @ 0x1e7d4c0] Increasing reorder buffer to 1
frame=  308 fps= 18 q=37.0 size=    1157kB time=00:00:10.84 bitrate= 873.9kbits/s
```

Let's access your converted file

- The nginx web server is installed on the cloud module, you can access your files by visiting the IP address of the cloud instance in your browser.



- If the page doesn't load then it's possible nginx isn't running, you can start nginx with the following command:
  - sudo /usr/local/nginx/sbin/nginx

More advanced options

- -ss <Position> -t <Duration>
  - Set the start position and duration of the output file (used to trim a file, e.g. –ss 00:00:05 –t 00:00:10 which trims from 5 – 15 seconds).
- -vf crop=<Width>:<Height>:<X>:<Y>
  - Crop a video dimensions starting at (X, Y) with the dimensions Width x Height
- -vf pad=<Width>:<Height>:<X>:<Y>:<Color>
  - Pad a video to fit a size staring at (X, Y) with the dimensions Width x Height with a color to fill the excess space (in hex).
- -frames:v <Frames>
  - The amount of video frames to output

Putting it to use

- Using a combination of the previous options, you're capable of performing some advanced operations.
- The below command grabs a thumbnail from a video at 5 seconds and saves it as thumb.jpg

```
ffmpeg -i input.mov -ss 00:00:05 -codec:v mjpeg -frames:v 1 -s
320x240 thumb.jpg
```

Try it out!

And there's more!

- This tutorial has only been a small glimpse of what's possible with FFMPEG. To view all the commands please visit:

  http://ffmpeg.org/ffmpeg.html

- You now have the ability to perform basic conversions via the command line. FFMPEG is used in production environments all over the web – try to do something interesting with it!

Thank you!

- Created by
  - Rishi Ishairzay (rishair@vt.edu)
  - Puloma De (pulomad@vt.edu)
  - Andrew Hwang (ajhwang@vt.edu)