Streaming

HLS

- We've seen how to host and play HLS videos
- Now we'll convert a .mp4 video into and HLS video
  - Using ffmpeg

ffmpeg

- Software with many video processing features
  - Needs to be installed on any machine running your server
  - Add the "ffmpeg" executable to your PATH variable
- Can use the command line to process videos

ffmpeg

```
ffmpeg -i inputVideo.avi outputVideo.mp4
```

- This is an example of very basic ffmpeg usage
- Use the -i flag to indicate the input filename
- The last argument is always the output filename
- This command will convert a video from avi to mp4

ffmpeg

```
ffmpeg –i inputVideo.avi –s 640x360 outputVideo.mp4
```

- We can add more parameters for the output file
  - Add flags and arguments before the output filename
  - Output filename is still the last argument
- This command will set the resolution of the output file to 640x360

ffmpeg

```
ffmpeg -i inputVideo.mp4 -f hls outputStream.m3u8
```

- ffmpeg will attempt to infer the type of the output file based on the file extension you provide
- However the m3u8 playlist file type is used by more that just HLS videos
- Need to specify the output format with the -f flag
- This is a minimal example of creating an HLS video

ffmpeg

- To run ffmpeg in your code
- Option 1: Make a system call
  - Same as typing a command into the command line
  - Build into every language
- Option 2: Download ffmpeg bindings for your language
  - Simplifies the syntax by calling methods instead of working with command line arguments
  - Makes the system calls for you

Live Streaming

- We've discussed streaming a video that has been uploaded as an mp4 (Can support any other format with ffmpeg)
  - Ex. YouTube
- How could we live stream?
  - Ex. Twitch

Live Streaming

- Use software that connects to a server via RTMP (Real Time Messaging Protocol)
  - OBS creates and sends an RTMP stream of video content
- Server continuously reads the video from the stream
- Typically use API keys to verify the user of the connection

Live Streaming

- Server converts the video into HLS
- Any client visiting the stream requests an m3u8 containing the latest files that have been processed
- When the client is running low on content, request an updated m3u8
  - Coded in the front-end JS of the site
- Server is constantly converting video and generating new playlists

Recall Buffering

- We have large file uploads
  - Don't submit your large files to AutoLab
- Remember how to buffer
  - Read content-length and read bytes until your read this many
  - Need to handle multipart POST requests as well
- Question for thought: What if a user uploads a file larger than your RAM?