Containers and codecs

- Containers
  - Check your work
- Codecs
  - Video
  - Audio

To support multiple browsers, you'll need to use FFmpeg to convert your `.mov` file to two different containers: an MP4 container and a WebM container. In actual practice, you would likely specify a codec at the same time. For now, we're letting FFmpeg use its defaults.

If these concepts are new to you, you should read Media file basics before going further.

Additionally, if you don't have FFmpeg installed read Media application basics to get it set up with Docker.

We are using the suggested Docker install and the glocken.mov file from Prepare media files for the web added inside the `media` directory. We used FFmpeg version 4.3.2 for all the commands in this section.

{% Aside 'caution' %} If the commands don't work for your version of FFmpeg, consult the FFmpeg documentation. {% endAside %}

Containers

First, we need to create our two containers from the `.mov` file with the `.mp4` and `.webm` file extensions with both an audio and video stream inside the file. Review Media file basics for more about containers and streams if you don't know the different between them.

1. MP4

   ```
   /media # ffmpeg -i glocken.mov glocken.mp4
   ```

2. WebM

```
/media # ffmpeg -i glocken.mov glocken.webm
```

WebM takes longer to create than MP4. This isn't surprising when you look at the results. While MP4 compresses to about `83%` of the original file's size, WebM is down to `78%` of the original's size, but can be much smaller. Your results will vary. It's important to call out that FFmpeg `4.2.2` set the default video bitrate to `200k` and in `4.3.2` it does not set a default bitrate. So the video is no longer **a mere `4%` of the original**. You can see this for yourself using the `ls -a` bash command in the folder where your media files are located.

For example:

```
/media # ls -l
-rw-r--r-- 1 root  root  12080306 Mar 7 12:16 glocken.mov
-rwx------ 1 root  root  10106446 Mar 7 12:33 glocken.mp4
-rwx------ 1 root  root   9503301 Mar 7 18:30 glocken.webm
```

To get a tiny file, you would do this instead:

```
/media # ffmpeg -i glocken.mov -b:v 200k glocken.webm
...
frame=  300 fps=3.6 q=0.0 Lsize=     483kB time=00:00:10.01 bitrate= 395.0kbits/s speed=0.121x
video:359kB audio:117kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead:
1.356068%
/media # ls -l
-rw-r--r-- 1 root  root  12080306 Mar 7 12:16 glocken.mov
-rwx------ 1 root  root  10106446 Mar 7 12:33 glocken.mp4
-rwx------ 1 root  root    494497 Mar 7 18:45 glocken.webm
```

Check your work

To verify your results, use FFmpeg and Shaka Packager as already shown in Media Application basics:

```
/media # packager input=glocken.mp4 --dump_stream_info
```

```
/media # ffmpeg -i glocken.mp4
```

Codecs

Next, the codec. As stated in Media file basics, a codec is *not* the same thing as a container (file type). Two files of the same container type could hold data compressed using different codecs. The WebM format for example allows audio to be encoded using either Vorbis or Opus. To change the codec we use FFmpeg. For example, this command outputs an `.mkv` file with a `vorbis` audio codec and an `av1` video codec.

```
/media # ffmpeg -i glocken.mov -c:a vorbis -c:v av1 glocken.mkv
```

In this example, the `-c:a` flag and the `-c:v` are for specifying the audio and video codecs respectively.

The Media conversion page lists commands needed to convert codecs. The tables below summarize the libraries used in FFmpeg to perform the codec conversions for WebM and MP4 files. These are the formats recommended for DASH and HLS respectively.

Video

Codec	Extension	Library
av1	WebM, mkv	libaom-av1
h264	MP4	libx264
vp9	WebM	libvpx-vp9

Audio

Codec	Extension	Library
aac	MP4	aac
opus	WebM	libopus
vorbis	WebM	libvorbis

Next, we'll show you how to change the bitrate of your newly created files.

□□□ #2
□ □□□ □□□ 2026-10-31 22:56:27 UTC
□ □□□ □□□ 2026-10-31 22:56:28 UTC