# Vertically or horizontally stack (mosaic) several videos using ffmpeg? [closed]

Asked 8 years, 7 months ago Active 6 months ago Viewed 40k times

41

38

**Closed.** This question does not meet Stack Overflow guidelines. It is not currently accepting answers.

**Want to improve this question?** Update the question so it's on-topic for Stack Overflow.

Closed 3 months ago.

Improve this question

I have two videos of the same exact length, and I would like to use ffmpeg to stack them into one video file.

How can I do this?

ffmpeg video-processing

Share Edit Follow Flag

edited Dec 11 '19 at 23:24
llogan
82.9k 21 159 185

asked Jul 19 '12 at 0:59
Joseph Turian
11.8k 12 41 61

## 3 Answers

Active Oldest Votes

26

See this answer to this question for a newer, simpler way to do this.

Old version:
You should be able to do this using the pad, movie and overlay filters in FFmpeg. The command will look something like this:

```
ffmpeg -i top.mov -vf 'pad=iw:2*ih [top]; movie=bottom.mov [bottom]; \
  [top][bottom] overlay=0:main_h/2' stacked.mov
```

First the movie that should be on top is padded to twice its height. Then the bottom movie is loaded. Then the bottom movie is overlaid on the padded top movie at an offset of half the padded movie's height.

Share Edit Follow Flag

edited Dec 13 '19 at 19:00

answered Jul 19 '12 at 2:32
blahdiblah
30.7k 18 91 149

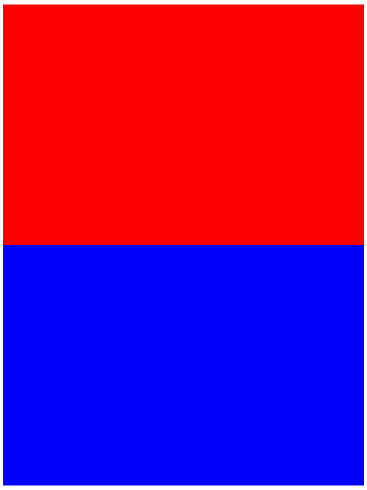
158

Use the vstack (vertical), hstack (horizontal), or xstack (custom layout) filters. It is easier and faster than other methods.

## Example 1: Combine/stack two videos or images

### Vertical

Using the vstack filter.

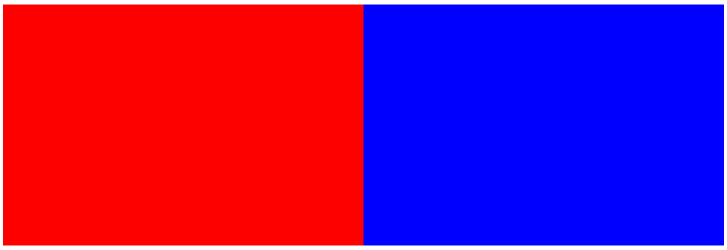


```
ffmpeg -i input0 -i input1 -filter_complex vstack=inputs=2 output
```

Videos must have the same width.

### Horizontal

Using the hstack filter.

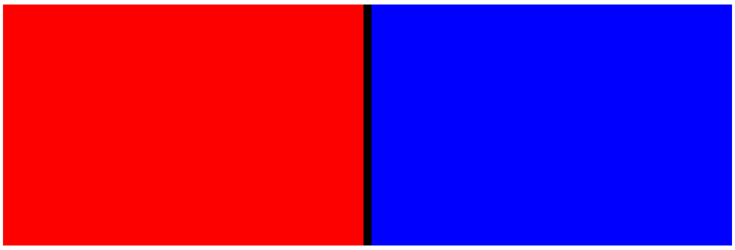


```
ffmpeg -i input0 -i input1 -filter_complex hstack=inputs=2 output
```

Videos must have the same height.

### With a border

Using the pad filter. This examples creates a 5px black border between the two sides.

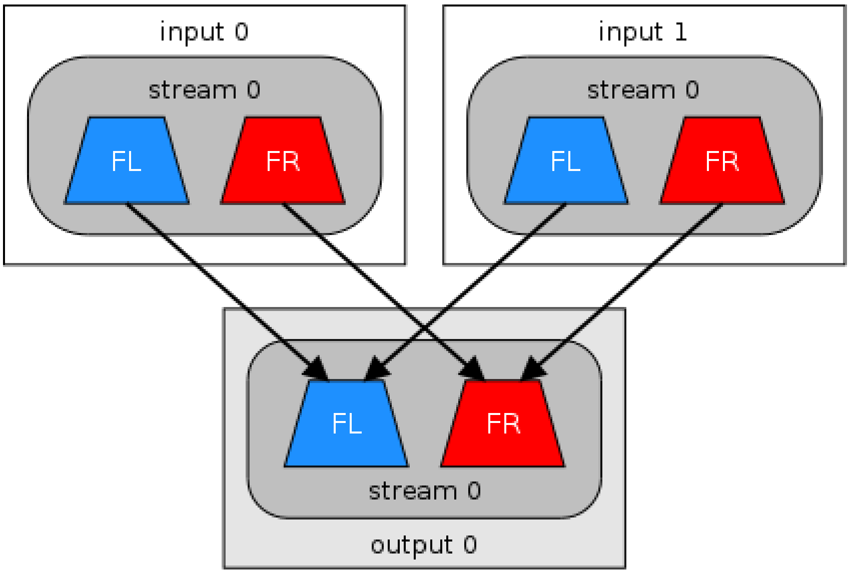


```
ffmpeg -i input0 -i input1 -filter_complex "[0]pad=iw+5:color=black[left];[left]
[1]hstack=inputs=2" ou
```

This answer is useful

## Example 2: Sam bove but with audio

### Downmix and use original channel placements

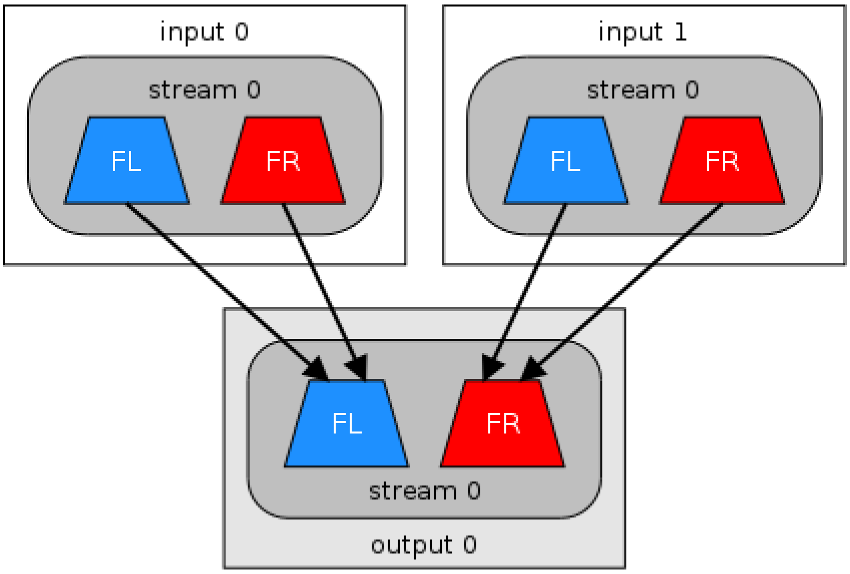


Add the amerge filter to combine the audio channels from both inputs:

```
ffmpeg -i input0 -i input1 -filter_complex "[0:v][1:v]vstack=inputs=2[v];[0:a]
[1:a]amerge=inputs=2[a]" -map "[v]" -map "[a]" -ac 2 output
```

- This assumes each input contains a stereo audio stream.
- `-ac 2` is included to downmix to stereo in case both inputs contain multi-channel audio. For example, if both inputs are stereo, you would get a 4-channel output audio stream instead of stereo if you omit `-ac 2`.

### Put all audio from each input into separate channels



Use amerge (or amix) and pan filters:

```
ffmpeg -i input0 -i input1 -filter_complex "[0:v][1:v]vstack=inputs=2[v];[0:a]
[1:a]amerge=inputs=2,pan=stereo|c0<c0+c1|c1<c2+c3[a]" -map "[v]" -map "[a]"  output
```

- This assumes each input contains a stereo audio stream.

### Using audio from one particular input

This example will use the audio from `input1`:

```
ffmpeg -i input0 -i input1 -filter_complex "[0:v][1:v]vstack=inputs=2[v]" -map "
[v]" -map 1:a output
```

### Adding silent audio / If one input does not have audio

If you mix inputs that have audio and inputs that do not have audio then amerge will fail because each input needs audio. You can add silent audio with the anullsrc filter to prevent this:

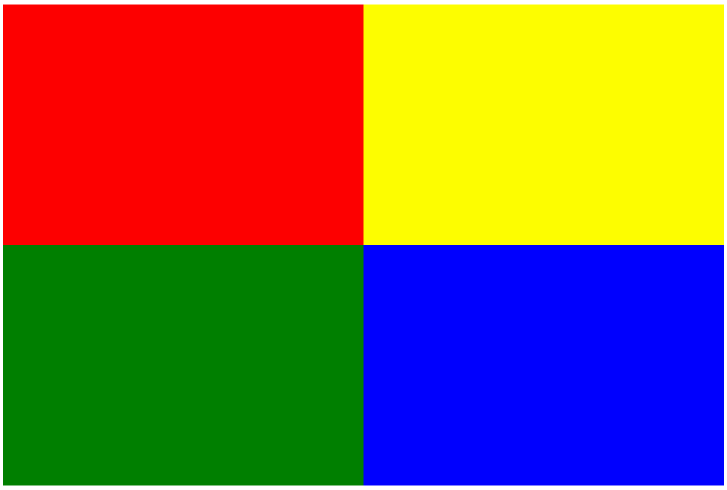
```
ffmpeg -i input0 -i input1 -filter_complex "[0:v]
[1:v]vstack=inputs=2[v];anullsrc[silent];[0:a][silent]amerge=inputs=2[a]" -map "
[v]" -map "[a]" -ac 2 output.mp4
```

## Example 3: Three videos or images

```
ffmpeg -i input0 -i input1 -i input2 -filter_complex "[0:v][1:v]
[2:v]hstack=inputs=3[v]" -map "[v]" output
```

If you want vertical use vstack instead of hstack.

## Example 4: 2x2 grid



### Using xstack

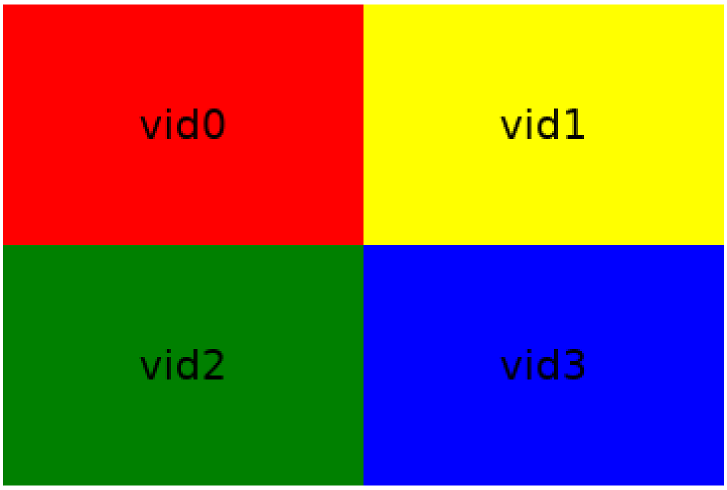
```
ffmpeg -i input0 -i input1 -i input2 -i input3 -filter_complex "[0:v][1:v][2:v]
[3:v]xstack=inputs=4:layout=0_0|w0_0|0_h0|w0_h0[v]" -map "[v]" output
```

### Using hstack and vstack

```
ffmpeg -i input0 -i input1 -i input2 -i input3 -filter_complex "[0:v]
[1:v]hstack=inputs=2[top];[2:v][3:v]hstack=inputs=2[bottom];[top]
[bottom]vstack=inputs=2[v]" -map "[v]" output
```

This syntax is easier to understand, but less efficient than using xstack as shown above.

## Example 5: 2x2 grid with text



Using the drawtext filter:

```
ffmpeg -i input0 -i input1 -i input2 -i input3 -filter_complex
"[0]drawtext=text='vid0':fontsize=20:x=(w-text_w)/2:y=(h-text_h)/2[v0];
 [1]drawtext=text='vid1':fontsize=20:x=(w-text_w)/2:y=(h-text_h)/2[v1];
 [2]drawtext=text='vid2':fontsize=20:x=(w-text_w)/2:y=(h-text_h)/2[v2];
 [3]drawtext=text='vid3':fontsize=20:x=(w-text_w)/2:y=(h-text_h)/2[v3];
 [v0][v1][v2][v3]xstack=inputs=4:layout=0_0|w0_0|0_h0|w0_h0[v]"
-map "[v]" output
```

## Example 6: Resize/scale an input

Since both videos need to have the same with for vstack, and the same height for hstack, you may need to scale one of the other videos to match the other:

Simple scale filter example to set width of input0 to 640 and automatically set height while preserving the aspect ratio:

```
ffmpeg -i input0 -i input2 -filter_complex "[0:v]scale=640:-1[v0];[v0]
[1:v]vstack=inputs=2" output
```

- For a more advanced method to fit any size video into a specific size while preserving aspect ratio see Resizing videos with ffmpeg to fit into static sized player.
- You can also use the scale2ref filter to automatically resize one video to match the dimensions of the other.

## Example 7: Delaying/pausing videos

This example will play the top left video while pausing the others. Once the top left video ends the top right video will play and so on.

Use the tpad, adelay, xstack, and amix filters:

```
ffmpeg -i top-left.mp4 -i top-right.mp4 -i bottom-left.mp4 -i bottom-right.mp4 -
filter_complex "[1]tpad=start_mode=clone:start_duration=5[tr];
[2]tpad=start_mode=clone:start_duration=10[bl];
[3]tpad=start_mode=clone:start_duration=15[br];[0][tr][bl]
[br]xstack=inputs=4:layout=0_0|w0_0|0_h0|w0_h0[v];[1:a]adelay=5s:all=true[a1];
[2:a]adelay=10s:all=true[a2];[3:a]adelay=15s:all=true[a3];[0:a][a1][a2]
[a3]amix=inputs=4[a]" -map "[v]" -map "[a]" output.mp4
```

- This example assumes each input is 5 seconds duration. Adjust `start_duration` and `adelay` values as needed.
- This command requires FFmpeg 4.2 or newer.
- If you don't like the complexity of xstack you can use several hstack/vstack instead as shown in **Example 4: 2x2 grid**.

Share Edit Follow Flag edited Aug 21 '20 at 18:51 answered Nov 17 '15 at 18:59

llogan
82.9k 21 159 185

1 `ffmpeg` reports `No such filter: 'vstack'` . Do I have to install it first? – frans Dec 12 '15 at 13:39

1 @frans You probably figured this out by now, but your `ffmpeg` is too old. Users are recommended to use a `ffmpeg` build derived from the current git master branch. – llogan Dec 14 '15 at 1:36

`Stream specifier ':a' in filtergraph description [0:v][1:v]vstack[v];[0:a][1:a]amerge=inputs=2[a] matches no streams.` Why do I get this? – RaduM Jun 29 '17 at 9:02

2 @RaduM its because one of your video doesnt have audio in it , you need to add an audio or silent audio in it , try the same command using two videos with Audio , it will work , am posting because I had the same problem today and rectified it using silent audio addition. – Jeffin Jul 10 '17 at 7:24

1 ffmpeg -i front.avi -i front.avi -i front.avi -i front.avi -i front.avi -i front.avi -filter_complex "[0:v][1:v][2:v]hstack=3[top];[3:v][4:v][5:v]hstack=3[bottom];[top][bottom]vstack[v]" -map "[v]" output.avi ............... but i cannot open the avi file – Kong Oct 11 '18 at 22:30

1

For 2 videos:

```
ffmpeg -i 1.mp4 -i 2.mp4 -filter_complex hstack out.mp4
```

For more videos(3 in this example):

```
ffmpeg -i 1.mp4 -i 2.mp4 -i 3.mp4 -filter_complex hstack=3 out.mp4
```

Share Edit Follow Flag

answered Apr 15 '20 at 10:36
mrgloom
14.7k 21 120 217