

Re-Live the Moment: Broadcasting Your Facebook Live Replay as a Live Stream on Instagram



[alex buzunov](#)

6 min read

May 18, 2025

4

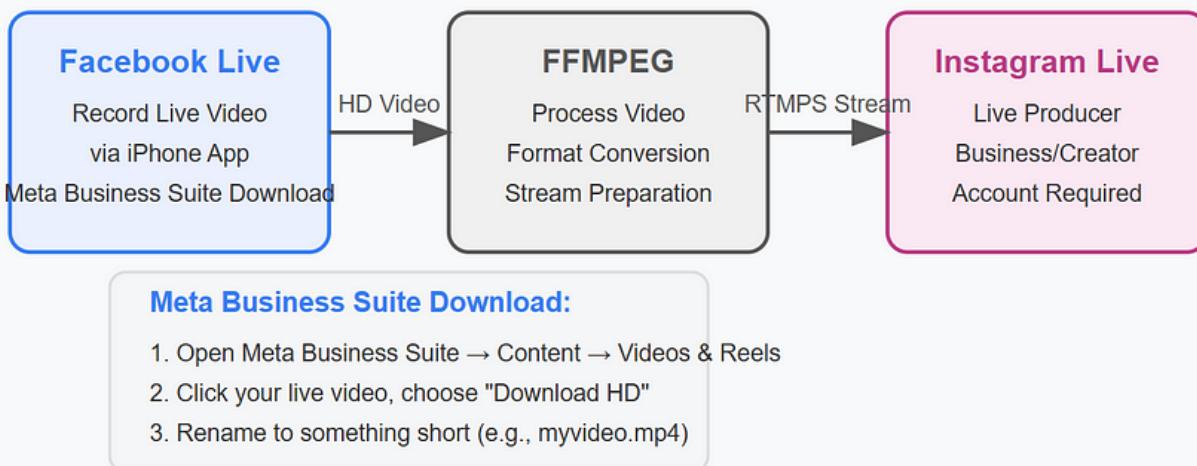
Ever capture a magical live stream only to wish you could show it to a whole new audience? With a little FFmpeg trickery, you can!

Why Re-Stream?

Live video triggers notifications, sits at the top of the feed, and generally owns a bigger slice of audience attention than a regular post. If you already have a polished Facebook Live session on disk, spinning

it back up on Instagram Live instantly doubles its mileage — no extra shooting, no extra editing, just extra reach.

Facebook Live to Instagram Streaming Workflow



Prerequisites:

- Facebook recorded Live video (download HD from Meta Business Suite)
- Instagram account with Live Producer enabled (Professional/Business account)
- FFMPEG installed (brew install ffmpeg for macOS, choco install ffmpeg for Windows)
- Your unique RTMPS URL & Stream Key from Instagram Live Producer
- Decent upload connection (~5 Mbps upstream for 1080p/30fps)

Path: Facebook Live → Meta Business Suite Download → FFMPEG Processing → Instagram Live Producer

Step 1 Download Your Facebook Live Video

1. Open **Meta Business Suite** and navigate to *Content ▷ Videos & Reels*.
2. Click your live video, choose **Download HD**.

3. Rename it something short — e.g. `myvideo.mp4`—and drop it in your streaming folder.

Tip: Trim awkward start/end silences with a quick `ffmpeg -i input.mp4 -ss 00:00:15 -to 00:45:00 -c copy trimmed.mp4`.

Alternative download

You can use [Facebook Video Downloader](#). It's handy wcPython script for video/reel download from Facebook.

The screenshot shows the Facebook Video Downloader application window. At the top, there is a 'Cookie File:' dropdown set to 'g\facebook_downloader\cookies.txt' with a 'Browse' button. The main area is a text log of a video download process:

```
ETA [0;33m00:00][0m
DEBUG: [download] [0;94m 81.0%[0m of 103.70MiB at [0;32m 31.48MiB/s[0m
ETA [0;33m00:00][0m
DEBUG: [download] [0;94m 84.9%[0m of 103.70MiB at [0;32m 31.69MiB/s[0m
ETA [0;33m00:00][0m
DEBUG: [download] [0;94m 88.7%[0m of 103.70MiB at [0;32m 32.00MiB/s[0m
ETA [0;33m00:00][0m
DEBUG: [download] [0;94m 92.6%[0m of 103.70MiB at [0;32m 32.10MiB/s[0m
ETA [0;33m00:00][0m
DEBUG: [download] [0;94m 96.4%[0m of 103.70MiB at [0;32m 32.41MiB/s[0m
ETA [0;33m00:00][0m
DEBUG: [download] [0;94m100.0%[0m of 103.70MiB at [0;32m 32.44MiB/s[0m
ETA [0;33m00:00][0m
DEBUG: [download] 100% of 103.70MiB in [1;37m00:00:04][0m at [0;32m25.69MiB/s[0m
Download successful. Result code: 0
Downloaded video: downloads\ArtForUkraine\d1524728-bb19-4bfe-acd8-4e1411a12e0a.mp4
Renaming downloads\ArtForUkraine\d1524728-bb19-4bfe-acd8-4e1411a12e0a.mp4
to downloads\ArtForUkraine\10ebb88d-ef4d-498b-8e2a-d9a19ed3c5f6_d1524728-
bb19-4bfe-acd8-4e1411a12e0a.mp4
```

At the bottom, there are four navigation buttons: 'Create', 'Reels', 'Backup', and 'Latest'.

Tip: You can also use it for reel download from YouTube/Instagram/Linkedin.

Step 2 Grab Your Instagram RTMPS Credentials

1. On desktop, head to **instagram.com** and click **Create** → **Live Video**.
2. In the right-hand pane choose **Streaming Software**.
3. Copy the **RTMPS URL** and **Stream Key**. Keep them secret!

You'll end up with something that looks like:

```
RTMPS URL: rtmps://edge-upload-lga3-1.xx.fbcnd.net:443/rtmp/  
Stream Key: 1810...e3-1&s_sw
```

Step 3 Fire Up FFmpeg

Paste the following Python helper into a new script or your terminal. It wraps the FFmpeg command so you only edit two variables: `video_path` and `stream_key`.

```
import subprocess  
  
# Path to your local video file  
video_path = 'myvideo.mp4' # Replace with your video path  
  
# Instagram RTMPS settings  
stream_url = 'rtmps://****-****-1.xx.fbcnd.net:443/rtmp/'  
stream_key = '***?s_bl=1&s_fbp=iad3-1&s_ow=10&s_prp=lga3-  
1&s_sw=0&s_tids=1&s_vt=ig&a=***' # Replace with actual key  
  
# Full RTMPS URL
```

```
rtmps_url = stream_url + stream_key

# FFmpeg command to stream the video
ffmpeg_command = [
    'ffmpeg',
    '-re', # Read input at native frame rate (simulate live)
    '-i', video_path, # Input file
    '-vcodec', 'libx264',
    '-preset', 'veryfast',
    '-maxrate', '4000k',
    '-bufsize', '8000k',
    '-pix_fmt', 'yuv420p',
    '-g', '50',
    '-acodec', 'aac',
    '-b:a', '128k',
    '-ar', '44100',
    '-f', 'flv',
    rtmps_url
]
# Run the command
subprocess.run(ffmpeg_command)
```

Parameter Cheatsheet

FFmpeg Parameter Cheatsheet

For Instagram Live Streaming

Example Command:

```
ffmpeg -re -i myvideo.mp4 -c:v libx264 -preset veryfast -maxrate 2500k  
-bufsize 5000k -g 60 -c:a aac -b:a 128k -pix_fmt yuv420p -f flv rtmps://...  
...
```

Flag	Meaning
<code>-re</code>	Throttles playback so FFmpeg doesn't sprint faster than realtime.
<code>-preset veryfast</code>	Encoder speed profile. <i>faster = higher CPU, lower quality</i>
<code>-maxrate / -bufsize</code>	Manage bitrate spikes to avoid buffering.
<code>-g</code>	Key-frame interval (GOP size). Ideal is <i>2 seconds × FPS</i> .
<code>-pix_fmt yuv420p</code>	The only pixel format Instagram accepts.



Tip: Use `-c:v libx264` for video and `-c:a aac` for audio encoding.

Troubleshooting

Troubleshooting Instagram Live Streams

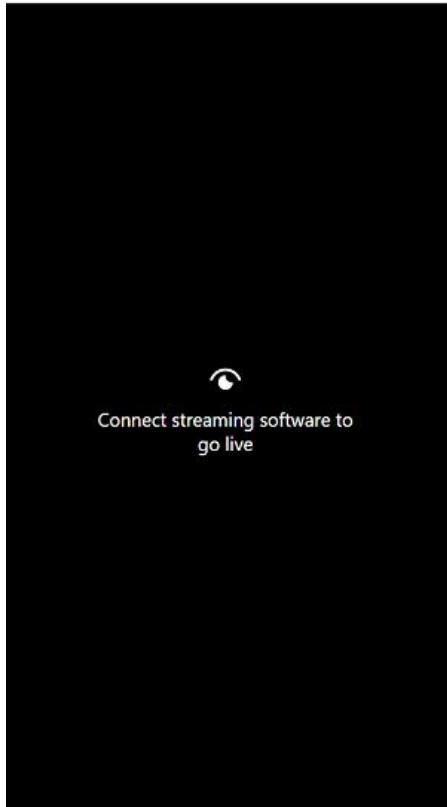
Symptom	Fix
Preview never appears	Double-check stream key, ensure port 443 outbound isn't blocked.
Audio out of sync	Add <code>-af "aresample=async=1"</code> or use CBR audio (<code>-b:a 160k -ar 48000</code>).
Choppy video	Drop <code>-maxrate</code> to <code>2500k</code> , switch preset to <code>ultrafast</code> , or lower resolution with <code>-vf scale=1280:720</code> .
Buffering loop	Your upload bandwidth is the bottleneck—stream at 720p/30 fps 2 Mbps.

 **Pro Tip:** Test your setup with `ffmpeg -loglevel debug` to see detailed error messages.

Go Live on Instagram

It's important to syncronise python script start and Instagram GoLive.
Do not let more than 15 sec delay.

You have to click on GoLive URL (top right corner).



Live video

Go live

Go live with streaming software

Copy and paste the stream key into your streaming software. This unique stream key tells your streaming software where to send your video feed and lets Instagram accept it.

Stream URL

rtmps://[REDACTED].ig-live.net:12345/[REDACTED]/rtmp/ [COPY](#)

Stream key

[REDACTED] [COPY](#)

How to stream

-  Copy the stream key and enter it into your streaming software.
-  Select Go live once your streaming software connects and displays.

The streaming software that's best for you depends on the type of content you plan to stream.

Learn more about [going live with streaming software](#).

After you go live

When you use the IG Live Producer tool, preview video is saved as part of the recording. If you want to share the recorded video, please download it and use Instagram's editing tools to cut out the preview section. Then share your video as a new post or reel.

Streaming Log

```
PS C:\Users\alex_\myg\golive> python .\go.py
ffmpeg version 6.1.1-essentials_build-www.gyan.dev Copyright (c) 2000-2023
the FFmpeg developers
  built with gcc 12.2.0 (Rev10, Built by MSYS2 project)
  configuration: --enable-gpl --enable-version3 --enable-static --pkg-
config=pkgconf --disable-w32threads --disable-autodetect --enable-fontconfig
--enable-iconv --enable-gnutls --enable-libxml2 --enable-gmp --enable-bzlib -
--enable-lzma --enable-zlib --enable-libsrt --enable-libssh --enable-libzmq --
--enable-avisynth --enable-sdl12 --enable-libwebp --enable-libx264 --enable-
libx265 --enable-libxvid --enable-libaom --enable-libopenjpeg --enable-libvpx
--enable-mediafoundation --enable-libass --enable-libfreetype --enable-
libfribidi --enable-libharfbuzz --enable-libvidstab --enable-libvmaf --
--enable-libzimg --enable-amf --enable-cuda-llvm --enable-cuvid --enable-
ffnvcodec --enable-nvdec --enable-nvenc --enable-dxva2 --enable-d3d11va --
--enable-libvpl --enable-libgme --enable-libopenmpt --enable-libopencore-amrwb
--enable-libmp3lame --enable-libtheora --enable-libvo-amrwbenc --enable-
libgsm --enable-libopencore-amrnb --enable-libopus --enable-libspeex --
--enable-libvorbis --enable-librubberband
  libavutil      58. 29.100 / 58. 29.100
```

```

libavcodec   60. 31.102 / 60. 31.102
libavformat  60. 16.100 / 60. 16.100
libavdevice  60. 3.100 / 60. 3.100
libavfilter   9. 12.100 / 9. 12.100
libswscale    7. 5.100 / 7. 5.100
libswresample 4. 12.100 / 4. 12.100
libpostproc   57. 3.100 / 57. 3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'myvideo.mp4':
Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
Duration: 00:25:20.17, start: 0.000000, bitrate: 572 kb/s
Stream #0:0[0x1] (und): Video: h264 (Main) (avc1 / 0x31637661), yuv420p(tv, unknown/bt709/bt709, progressive), 432x768, 501 kb/s, 29.99 fps, 30 tbr, 16k tbn (default)
Metadata:
    handler_name    : VideoHandler
    vendor_id       : [0][0][0][0]
Stream #0:1[0x2] (und): Audio: aac (HE-AAC) (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 64 kb/s (default)
Metadata:
    handler_name    : SoundHandler
    vendor_id       : [0][0][0][0]
Stream mapping:
Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
Stream #0:1 -> #0:1 (aac (native) -> aac (native))
Press [q] to stop, [?] for help
[libx264 @ 000001a5113eeac0] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
[libx264 @ 000001a5113eeac0] profile High, level 3.0, 4:2:0, 8-bit
[libx264 @ 000001a5113eeac0] 264 - core 164 r3172 c1c9931 - H.264/MPEG-4 AVC codec - Copyleft 2003-2023 - http://www.videolan.org/x264.html - options:
cabac=1 ref=1 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=2 psy=1
psy_rd=1.00:0.00 mixed_ref=0 me_range=16 chroma_me=1 trellis=0 8x8dct=1 cqm=0
deadzone=21,11 fast_pskip=1 chroma_qp_offset=0 threads=24 lookahead_threads=6
sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0
constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1
weightb=1 open_gop=0 weightp=1 keyint=50 keyint_min=5 scenecut=40
intra_refresh=0 rc_lookahead=10 rc=crf mbtree=1 crf=23.0 qcomp=0.60 qpmin=0
qpmax=69 qpstep=4 vbv_maxrate=4000 vbv_bufsize=8000 crf_max=0.0 nal_hrd=none
filler=0 ip_ratio=1.40 aq=1:1.00
Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf60.16.100
Stream #0:0(und): Video: h264 ([7][0][0][0] / 0x0007), yuv420p(tv, unknown/bt709/bt709, progressive), 432x768, q=2-31, 30 fps, 1k tbn (default)
Metadata:
    handler_name    : VideoHandler
    vendor_id       : [0][0][0][0]
    encoder         : Lavc60.31.102 libx264
Side data:
```

```

cpb: bitrate max/min/avg: 4000000/0/0 buffer size: 8000000 vbv_delay:
N/A
Stream #0:1(und): Audio: aac (LC) ([10][0][0][0] / 0x000A), 44100 Hz,
stereo, fltp, 128 kb/s (default)
Metadata:
    handler_name      : SoundHandler
    vendor_id         : [0][0][0][0]
    encoder           : Lavc60.31.102 aac
[flv @ 000001a5117bb900] Failed to update header with correct
duration.7.6kbits/s speed= 1x
[flv @ 000001a5117bb900] Failed to update header with correct filesize.
[out#0/flv @ 000001a511333a00] video:191106kB audio:23822kB subtitle:0kB
other streams:0kB global headers:0kB muxing overhead: 0.920159%
frame=45584 fps= 30 q=-1.0 Lsize= 216906kB time=00:25:20.12
bitrate=1168.9kbytes/s speed= 1x
[libx264 @ 000001a5113eeac0] frame I:929 Avg QP:22.21 size: 29503
[libx264 @ 000001a5113eeac0] frame P:19964 Avg QP:25.22 size: 6476
[libx264 @ 000001a5113eeac0] frame B:24691 Avg QP:28.87 size: 1579
[libx264 @ 000001a5113eeac0] consecutive B-frames: 13.5% 34.2% 26.3% 26.1%
[libx264 @ 000001a5113eeac0] mb I I16..4: 7.7% 32.8% 59.5%
[libx264 @ 000001a5113eeac0] mb P I16..4: 4.6% 11.1% 2.9% P16..4: 33.1%
15.1% 9.4% 0.0% 0.0% skip:23.8%
[libx264 @ 000001a5113eeac0] mb B I16..4: 1.0% 1.2% 0.1% B16..8: 22.3%
6.7% 0.8% direct: 4.2% skip:63.8% I0:30.4% I1:56.8% BI:12.8%
[libx264 @ 000001a5113eeac0] 8x8 transform intra:54.0% inter:25.2%
[libx264 @ 000001a5113eeac0] coded y,uvDC,uvAC intra: 57.0% 50.0% 15.0%
inter: 9.4% 8.5% 0.4%
[libx264 @ 000001a5113eeac0] i16 v,h,dc,p: 35% 24% 27% 14%
[libx264 @ 000001a5113eeac0] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 34% 24% 19% 3%
3% 4% 4% 5%
[libx264 @ 000001a5113eeac0] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 27% 20% 13% 6%
7% 7% 7% 6%
[libx264 @ 000001a5113eeac0] i8c dc,h,v,p: 59% 16% 20% 5%
[libx264 @ 000001a5113eeac0] Weighted P-Frames: Y:7.2% UV:1.8%
[libx264 @ 000001a5113eeac0] kb/s:1029.84
[aac @ 000001a5113e6600] Qavg: 892.209

```

Wrapping Up

And that's it! Three quick steps and you've breathed new life into a past performance. Re-streaming lets you build momentum on Instagram without rehearsals, re-shoots, or extra gear — just code, coffee, and creativity.

*Ready to hit **Go Live** again? Tag me [@alex.bzv.art](#) when you do — I'd love to tune in!*

Next Up

-  Automating multi-platform live restreams with Node & OBS WebSocket
-  Designing vertical-first overlays for Instagram Live
-  Analyzing viewer retention with Instagram Insights & Python

Facebook

Instagram

Live Streaming

Python

Meta Business Suite