CONTENTS

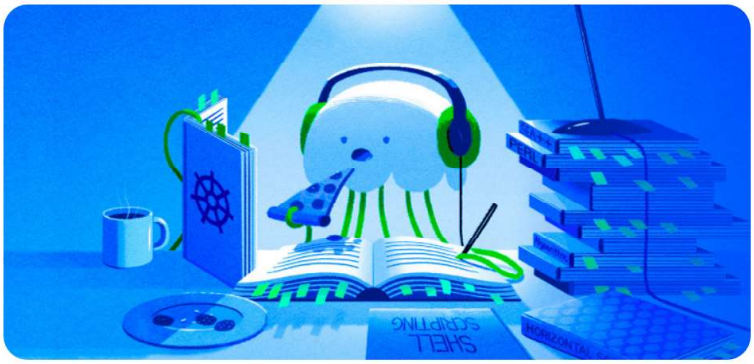
// Tutorial //

How To Set Up a Video Streaming Server using Nginx-RTMP on Ubuntu 20.04

Published on January 7, 2022

Ubuntu Ubuntu 20.04 Nginx

By Alex Garnett
Senior DevOps Technical Writer



Not using Ubuntu 20.04?
Choose a different version or distribution.

Ubuntu 20.04

Introduction

There are many use cases for streaming video. Service providers such as Twitch are very popular for handling the web discovery and community management aspects of streaming, and free software such as OBS Studio is widely used for combining video overlays from multiple different stream sources in real time. While these platforms are very powerful, in some cases you may want to be able to host a stream that does not rely on other service providers.

In this tutorial, you will learn how to configure the Nginx web server to host an independent RTMP video stream that can be linked and viewed in different applications. RTMP, the Real-Time Messaging Protocol, defines the fundamentals of most internet video streaming. You will also learn how to host HLS and DASH streams that support more modern platforms using the same technology.

Prerequisites

To complete this guide, you will need:

- An Ubuntu 20.04 server and a non-root user with sudo privileges. You can learn more about how to set up a user with these privileges in our Initial Server Setup with Ubuntu 20.04 guide.
- Nginx installed, following How To Install Nginx on Ubuntu 20.04.

This tutorial will use the placeholder domain name **your_domain** for URLs and hostnames. Substitute this with your own domain name or IP address as you work through the tutorial.

Step 1 – Installing and Configuring Nginx-RTMP

Most modern streaming tools support the RTMP protocol, which defines the basic parameters of an internet video stream. The Nginx web server includes a module that allows you to provide an RTMP stream with minimal configuration from a dedicated URL, just like it provides HTTP access to web pages by default. The Nginx RTMP module isn't included automatically with Nginx, but on Ubuntu 20.04 and most other Linux distributions you can install it as an additional package.

Begin by running the following commands as a non-root user to update your package listings and install the Nginx module:

```
$ sudo apt update
$ sudo apt install libnginx-mod-rtmp
```

Copy

Installing the module won't automatically start providing a stream. You'll need to add a configuration block to your Nginx configuration file that defines where and how the stream will be available.

Using `nano` or your favorite text editor, open Nginx's main configuration file, `/etc/nginx/nginx.conf`, and add this configuration block to the end of the file:

```
$ sudo nano /etc/nginx/nginx.conf
```
Copy

/etc/nginx/nginx.conf

```
. . .
rtmp {
        server {
                listen 1935;
                chunk_size 4096;
                allow publish 127.0.0.1;
                deny publish all;

                application live {
                        live on;
                        record off;
                }
        }
}
```

- `listen 1935` means that RTMP will be listening for connections on port 1935, which is standard.
- `chunk_size 4096` means that RTMP will be sending data in 4KB blocks, which is also standard.
- `allow publish 127.0.0.1` and `deny publish all` mean that the server will only allow video to be published from the same server, to avoid any other users pushing their own streams.
- `application live` defines an application block that will be available at the `/live` URL path.
- `live on` enables live mode so that multiple users can connect to your stream concurrently, a baseline assumption of video streaming.
- `record off` disables Nginx-RTMP's recording functionality, so that all streams are not separately saved to disk by default.

Save and close the file. If you are using `nano`, press `Ctrl+X`, then when prompted, `Y` and Enter.

This provides the beginning of your RTMP configuration. By default, it listens on port `1935`, which means you'll need to open that port in your firewall. If you configured `ufw` as part of your initial server setup run the following command.

```
$ sudo ufw allow 1935/tcp
```
Copy

Now you can reload Nginx with your changes:

```
$ sudo systemctl reload nginx.service
```
Copy

You should now have a working RTMP server. In the next section, we'll cover streaming video to your RTMP server from both local and remote sources.

Step 2 – Sending Video to Your RTMP Server

There are multiple ways to send video to your RTMP server. One option is to use `ffmpeg`, a popular command line audio-video utility, to play a video file directly on your server. If you don't have a video file already on the server, you can download one using `youtube-dl`, a command line tool for capturing video from streaming platforms like YouTube. In order to use `youtube-dl`, you'll need an up to date Python installation on your server as well.

First, install Python and its package manager, `pip`:

```
$ sudo apt install python3-pip
```
Copy

Next, use `pip` to install `youtube-dl`:

```
$ sudo pip install youtube-dl
```
Copy

Now you can use `youtube-dl` to download a video from YouTube. If you don't have one in mind, try this video, introducing DigitalOcean's App Platform:

```
$ youtube-dl https://www.youtube.com/watch?v=iom_nhYQIYk
```
Copy

You'll see some output as `youtube-dl` combines the video and audio streams it's downloading back into a single file – this is normal.

```
Output
[youtube] iom_nhYQIYk: Downloading webpage
WARNING: Requested formats are incompatible for merge and will be merged into mkv.
[download] Destination: Introducing App Platform by DigitalOcean-iom nhYQIYk.f137.mp4
[download] 100% of 32.82MiB in 08:40
[download] Destination: Introducing App Platform by DigitalOcean-iom_nhYQIYk.f251.webm
[download] 100% of 1.94MiB in 00:38
[ffmpeg] Merging formats into "Introducing App Platform by DigitalOcean-iom_nhYQIYk.mkv"
Deleting original file Introducing App Platform by DigitalOcean-iom_nhYQIYk.f137.mp4 (pass -k to
Deleting original file Introducing App Platform by DigitalOcean-iom_nhYQIYk.f251.webm (pass -k t
```

You should now have a video file in your current directory with a title like `Introducing App Platform by DigitalOcean-iom_nhYQIYk.mkv`. In order to stream it, you'll want to install `ffmpeg`:

```
$ sudo apt install ffmpeg
```

Copy

And use `ffmpeg` to send it to your RTMP server:

```
$ ffmpeg -re -i "Introducing App Platform by DigitalOcean-iom_nhYQIYk.mkv" -c:v copy -c
```

Copy

This `ffmpeg` command is doing a few things to prepare the video for a streaming-friendly format. This isn't an `ffmpeg` tutorial, so you don't need to examine it too closely, but you can understand the various options as follows:

- `-re` specifies that input will be read at its native framerate.
- `-i "Introducing App Platform by DigitalOcean-iom_nhYQIYk.mkv"` specifies the path to our **i**nput file.
- `-c:v` is set to `copy`, meaning that you're copying over the **v**ideo format you got from YouTube natively.
- `-c:a` has other parameters, namely `aac -ar 44100 -ac 1`, because you need to resample the **a**udio to an RTMP-friendly format. `aac` is a widely supported audio codec, `44100` hz is a common frequency, and `-ac 1` specifies the first version of the AAC spec for compatibility purposes.
- `-f flv` wraps the video in an `flv` format container for maximum compatibility with RTMP.

The video is sent to `rtmp://localhost/live/stream` because you defined the `live` configuration block in Step 1, and `stream` is an arbitrarily chosen URL for this video.

Note: You can learn more about `ffmpeg` options from ffmprovisr, a community-maintained catalog of `ffmpeg` command examples, or refer to the official documentation.

While `ffmpeg` is streaming the video, it will print timecodes:

```
Output
frame=  127 fps= 25 q=-1.0 size=     405kB time=00:00:05.00 bitrate= 662.2kbits/s speed=frame=
```

This is standard `ffmpeg` output. If you were converting video to a different format, these might be helpful in order to understand how efficiently the video is being resampled, but in this case, you just want to see that it's being played back consistently. Using this sample video, you should get exact `fps= 25` increments.

While ffmpeg is running, you can connect to your RTMP stream from a video player. If you have `vlc`, `mpv`, or another media player installed locally, you should be able to view your stream by opening the URL `rtmp://your_domain/live/stream` in your media player. Your stream will terminate after `ffmpeg` has finished playing the video. If you want it to keep looping indefinitely, you can add `-stream_loop -1` to the beginning of your `ffmpeg` command.

Note: You can also stream directly to, for example, Facebook Live using `ffmpeg` without needing to use Nginx-RTMP at all by replacing `rtmp://localhost/live/stream` in your `ffmpeg` command with `rtmps://live-api-s.facebook.com:443/rtmp/your-facebook-stream-key`. YouTube uses URLs like `rtmp://a.rtmp.youtube.com/live2`. Other streaming providers that can consume RTMP streams should behave similarly.

Now that you've learned to stream static video sources from the command line, you'll learn how to stream video from dynamic sources using OBS on a desktop.

Step 3 – Streaming Video to Your Server via OBS (Optional)

Streaming via `ffmpeg` is convenient when you have a prepared video that you want to play back, but live streaming can be much more dynamic. The most popular software for live streaming is OBS, or Open Broadcaster Software – it is free, open source, and very powerful.

OBS is a desktop application, and will connect to your server from your local computer.

After installing OBS, configuring it means customizing which of your desktop windows and audio sources you want to add to your stream, and then adding credentials for a streaming service. This tutorial will not be covering your streaming configuration, as it is down to preference, and by default, you can have a working demo by just streaming your entire desktop. In order to set your streaming service credentials, open OBS' settings menu, navigate to the *Stream* option and input the following options:

```
Streaming Service: Custom
Server: rtmp://your_domain/live
Play Path/Stream Key: obs_stream
```

`obs_stream` is an arbitrarily chosen path – in this case, your video would be available at `rtmp://your_domain/live/obs_stream`. You do not need to enable authentication, but you do need to add an additional entry to the IP whitelist that you configured in Step 1.

Back on the server, open Nginx's main configuration file, `/etc/nginx/nginx.conf`, and add an additional `allow publish` entry for your local IP address. If you don't know your local IP address, it's best to just go to a site like What's my IP which can tell you where you accessed it from:

```
$ sudo nano /etc/nginx/nginx.conf
```
Copy

/etc/nginx/nginx.conf

```
. . .
                allow publish 127.0.0.1;
                allow publish your_local_ip_address;
                deny publish all;
. . .
```

Save and close the file, then reload Nginx:

```
$ sudo systemctl reload nginx.service
```
Copy

You should now be able to close OBS' settings menu and click `Start Streaming` from the main interface! Try viewing your stream in a media player as before. Now that you've seen the fundamentals of streaming video in action, you can add a few other features to your server to make it more production-ready.

Step 4 – Adding Monitoring to Your Configuration (Optional)

Now that you have Nginx configured to stream video using the Nginx-RTMP module, a common next step is to enable the RTMP statistics page. Rather than adding more and more configuration details to your main `nginx.conf` file, Nginx allows you to add per-site configurations to individual files in a subdirectory called `sites-available/`. In this case, you'll create one called `rtmp`:

```
$ sudo nano /etc/nginx/sites-available/rtmp
```
Copy

Add the following contents:

/etc/nginx/sites-available/rtmp

```
server {
    listen 8080;
    server_name  localhost;

    # rtmp stat
    location /stat {
        rtmp_stat all;
        rtmp_stat_stylesheet stat.xsl;
    }
    location /stat.xsl {
        root /var/www/html/rtmp;
    }

    # rtmp control
    location /control {
        rtmp control all;
    }
}
```

Save and close the file. The `stat.xsl` file from this configuration block is used to style and display an RTMP statistics page in your browser. It is provided by the `libnginx-mod-rtmp` library that you installed earlier, but it comes zipped up by default, so you will need to unzip it and put it in the `/var/www/html/rtmp` directory to match the above configuration. Note that you can find additional information about any of these options in the Nginx-RTMP documentation.

Create the `/var/www/html/rtmp` directory, and then uncompress the `stat.xsl.gz` file with the following commands:

```
$ sudo mkdir /var/www/html/rtmp
$ sudo gunzip -c /usr/share/doc/libnginx-mod-rtmp/examples/stat.xsl.gz > /var/www/html/rtmp/sta
```
Copy

Finally, to access the statistics page that you added, you will need to open another port in your firewall. Specifically, the `listen` directive is configured with port `8080`, so you will need to add a rule to access Nginx on that port. However, you probably don't want others to be able to access your stats page, so it's best only to allow it for your own IP address. Run the following command:

```
$ sudo ufw allow from your_ip_address to any port http-alt
```
Copy

Next, you'll need to activate this new configuration. Nginx's convention is to create symbolic links (like shortcuts) from files in `sites-available/` to another folder called `sites-enabled/` as you decide to enable or disable them. Using full paths for clarity, make that link:

```
$ sudo ln -s /etc/nginx/sites-available/rtmp /etc/nginx/sites-enabled/rtmp
```
Copy

Now you can reload Nginx again to process your changes:

```
$ sudo systemctl reload nginx.service
```
Copy

You should now be able to go to `http://your_domain:8080/stat` in a browser to see the RTMP statistics page. Visit and refresh the page while streaming video and watch as the stream statistics change.

You've now seen how to monitor your video stream and push it to third party providers. In the final section, you'll learn how to provide it directly in a browser without the use of third party streaming platforms or standalone media player apps.

Step 5 – Creating Modern Streams for Browsers (Optional)

As a final step, you may want to add support for newer streaming protocols so that users can stream video from your server using a web browser directly. There are two protocols that you can use to create HTTP-based video streams: Apple's HLS and MPEG DASH. They both have advantages and disadvantages, so you will probably want to support both.

The Nginx-RTMP module supports both standards. To add HLS and DASH support to your server, you will need to modify the `rtmp` block in your `nginx.conf` file. Open `/etc/nginx/nginx.conf` using `nano` or your preferred editor, then add the following highlighted directives:

```
$ sudo nano /etc/nginx/nginx.conf
```
Copy

/etc/nginx/nginx.conf

```
. . .
rtmp {
        server {
. . .
                application live {
                        live on;
                        record off;
                        hls on;
                        hls_path /var/www/html/stream/hls;
                        hls_fragment 3;
                        hls_playlist_length 60;

                        dash on;
                        dash_path /var/www/html/stream/dash;
                }
        }
}
. . .
```

Save and close the file. Next, add this to the bottom of your `sites-available/rtmp`:

```
$ sudo nano /etc/nginx/sites-available/rtmp
```
Copy

/etc/nginx/sites-available/rtmp

```
. . .
server {
    listen 8088;

    location / {
        add_header Access-Control-Allow-Origin *;
        root /var/www/html/stream;
    }
}

types {
    application/dash+xml mpd;
}
```

Note: The `Access-Control-Allow-Origin *` header enables *CORS*, or Cross-Origin Resource Sharing, which is disabled by default. This communicates to any web browsers accessing data from your server that the server may load resources from other ports or domains. CORS is needed for maximum compatibility with HLS and DASH clients, and a common configuration toggle in many other web deployments.

Save and close the file. Note that you're using port `8088` here, which is another arbitrary choice for this tutorial to avoid conflicting with any services you may be running on port 80 or 443. You'll want to open that port in your firewall for now too:

```
$ sudo ufw allow 8088/tcp
```
Copy

Finally, create a `stream` directory in your web root to match the configuration block, so that Nginx can generate the necessary files for HLS and DASH:

```
$ sudo mkdir /var/www/html/stream
```
Copy

Reload Nginx again:

```
$ sudo systemctl reload nginx
```
Copy

You should now have an HLS stream available at `http://your_domain:8088/hls/stream.m3u8` and a DASH stream available at `http://your_domain:8088/dash/stream.mpd`. These endpoints will generate any necessary metadata on top of your RTMP video feed in order to support modern APIs.

Conclusion

The configuration options that you used in this tutorial are all documented in the Nginx RTMP Wiki

page. Nginx modules typically share common syntax and expose a very large set of configuration options, and you can review their documentation to change any of your settings from here.

Nearly all internet video streaming is implemented on top of RTMP, HLS, and DASH, and by using the approach that you have explored in this tutorial, you can provide your stream via other broadcasting services, or expose it any other way you choose. Next, you could look into configuring Nginx as a reverse proxy in order to make some of these different video endpoints available as subdomains.

About the authors

Alex Garnett Author
Senior DevOps Technical Writer

Comments

9 Comments

1552f1d52be248b3998a8370661c48 • March 14, 2023

thanks! i can play my obs stream on vlc player with rtmp://{domain_name}/live url. but i have a question: how to add https protocols to the hls or dash?

Reply

woods.marcus • October 31, 2022

Thank you so much for the clear and excellent article. I was able to get this done minus the following issue below!

As of today, 10/31/2022 the following command does not work properly. The file does not exist when I try to run it. I ended up getting the xsl file from the github repo. I did do a find on the system and was unable to find any stat.xsl.gz available, so it appears they may not package this file any longer with the rtmp module. Please update accordingly or let me know if I am wrong.

```
sudo gunzip -c /usr/share/doc/libnginx-mod-rtmp/examples/stat.xsl.gz > /var/www/html/rt
```

How To Set Up a Video Streaming Server using Nginx-RTMP on Ubuntu 20.04

Reply