



Resources ▸ Blog ▸ Tech

Enabling Video Streaming for Remote Learning with NGINX and NGINX Plus



Nina Forsyth of F5

Sr. Product Manager | April 16, 2020

I was thrilled when F5 announced the [purchase of NGINX](#) a year ago, because as a longtime F5er, I knew NGINX was a perfect fit. That feeling was confirmed when, in one of our first joint meetings, NGINX employees discussed something deeply important to me: their commitment to the “community”. Since joining the NGINX team as a product manager, I’ve had the pleasure of seeing that commitment in action and gotten to work more deeply with the community.

The NGINX Community Is Coming Together During the COVID-19 Pandemic

Those of us working with open source typically use “community” to describe people who support an open source project by submitting code. While important, they are only a small part of the NGINX community. Who, then, are the other members of the community? We also value those of you who:

- Use NGINX in the projects you share on GitHub
- Rely on NGINX to securely deliver sites, apps, and APIs to your users
- Write blog posts on installing and using NGINX
- Help NGINX users on public forums

In recent months, members of the NGINX community have truly come together to support each other and the many organizations struggling to maintain operations during the COVID-19 pandemic. Many of you have flagged sites that might benefit from NGINX software, or even donated time and expertise on how to tune NGINX for optimal performance. In an effort to do our part, we’ve published [Free Resources for Websites Impacted by COVID-19](#), and have been excited to see the community rally around new NGINX users.

As part of our efforts to help organizations cope with increased traffic and add new functionality, we’ve been providing up to five free instances of NGINX Plus to organizations on the frontline of the pandemic. That has included medical providers, government offices, and – most relevant to this blog – schools, universities, and education-focused non-profits. We detected a trending use case among these organizations that probably won’t surprise you: **streaming video!**


With schools around the world shutting their doors and rapidly implementing distance learning, we expect the use of streaming video to increase exponentially over the coming weeks and months. In the rest of this blog I explain how you can use NGINX Open Source and NGINX Plus to enable stable, secure, and scalable video streaming.

A big thanks goes out to NGINX Technical Solutions Architect James Jones for recording demo videos to accompany this blog!

Streaming Live Video and Storing Videos with NGINX Open Source

Our solution for streaming video takes advantage of the Real-Time Messaging Protocol (RTMP) module for NGINX. In this video, James goes through the process step by step:





So that you don't have to take notes while watching the James's demo, we've captured all of the commands and configuration in the following sections.

- [Installing the Build Tools](#)
- [Installing Dependencies](#)
- [Compiling NGINX with the RTMP Module](#)
- [Configuring NGINX](#)
- [Validating the Configuration and Starting NGINX](#)
- [Testing the Playback Methods](#)

An Important Note: Don't Forget Security!

The NGINX configurations presented in this blog do not include security measures to restrict who can watch your video stream. There are a variety of ways to secure your streams with the front-end application your viewers use to watch the video, such as allowing access only from certain IP addresses or requiring viewers to authenticate.

Installing the Build Tools

Before compiling NGINX, you need to have some basic build tools installed: `autoconf`, `gcc`, `git`, and `make`. To download and install them, run the command for your operating system (if it's not included here, consult the OS vendor documentation).

- For Debian and Ubuntu:

```
$ sudo apt update
$ sudo apt install build-essential git
```

- For CentOS, Oracle Linux, and RHEL:

```
$ sudo yum update
$ sudo yum groupinstall "Development Tools"
$ sudo yum install git
```

Installing Dependencies

The NGINX build also requires several dependencies: Perl Compatible Regular Expressions (PCRE), OpenSSL, and zlib for compression.

Installing Dependencies with a Package Manager

The easier way to download and install the dependencies is with a package manager. Run the command for your operating system (if it's not included here, consult the OS vendor documentation).

- For Debian and Ubuntu:

```
$ sudo apt install libpcre3-dev libssl-dev zlib1g-dev
```

- For CentOS, Oracle Linux, and RHEL:

```
$ sudo yum groupinstall pcre-devel zlib-devel openssl-devel
```

Installing Dependencies from Source

If you instead want to build and install the dependencies from source, see our [instructions](#).

Compiling NGINX with the RTMP Module

To complete the build, you clone the GitHub repositories for RTMP and NGINX, run the NGINX `configure` command, and then compile NGINX.

```
$ cd /path/to/build/dir
$ git clone https://github.com/arut/nginx-rtmp-module.git
$ git clone https://github.com/nginx/nginx.git
$ cd nginx
$ ./auto/configure --add-module=../nginx-rtmp-module
$ make
$ sudo make install
```

Configuring NGINX

You can configure NGINX to stream video using one or both of the HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) protocols. The protocols provide the same functionality, so choosing between them is really a matter of preference. If you're not familiar with them, see [HLS vs DASH](#) on the Viddeo blog.

HLS Configuration

For HLS, the configuration is as follows. In the demo (at time point 5:10), James explains the purpose of these directives.

```
rtmp {
    server {
        listen 1935;
        application live {
            live on;
            interleave on;

            hls on;
            hls_path /tmp/hls;
            hls_fragment 15s;
        }
    }
}

http {
    default_type application/octet-stream;

    server {
        listen 80;
        location /tv {
            root /tmp/hls;
        }
    }

    types {
        application/vnd.apple.mpegurl m3u8;
        video/mp2t ts;
        text/html html;
    }
}
```

DASH Configuration

For DASH, the configuration is as follows. In the demo, James combines HLS and DASH in a single configuration, as many directives are the same for both protocols.

```
rtmp {
    server {
        listen 1935;
        application live {
            live on;
            dash on;
            dash_path /tmp/dash;
            dash_fragment 15s;
        }
    }
}

http {
    server {
        listen 80;
        location /tv {
            root /tmp/dash;
        }
    }

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

Validating the Configuration and Starting NGINX

It's always a good idea to validate your NGINX configuration to make sure there are no syntactic errors. Run this command:

```
$ sudo nginx -t
nginx: the configuration file filename syntax is ok
nginx: configuration file filename test is successful
```

Then run this command to start NGINX:

```
$ sudo nginx
```

Testing the Playback Methods

Start your video stream. **OBS Studio** is a commonly used open source tool that allows you to livestream from your workstation to your NGINX server by configuring a custom RTMP server. Configure OBS to stream to `rtmp://NGINX_server/tv/tv2`, where `NGINX_server` is the IP address or hostname of your NGINX server. No stream key is required.

James doesn't use OBS in the demo because he is streaming video from a file rather than live. He starts the video stream (at 9:30) by running the `stream.sh` script, which has these contents:

```
ffmpeg -re -i bbb_sunflower_1080p_60fps_normal.mp4 -vcodec copy -loop -1 -c:a aac -b:a 160k -ar 44100 -strict -2 -f flv rtmp:192.168.1.138/live/bbb
```

The video he is streaming (specified with the `-i` argument) is the open source **Big Buck Bunny** video from [blender.org](#). For details about the other arguments, see the [ffmpeg documentation](#).

Once video is streaming, you can test that NGINX is correctly serving it using the protocols you have configured. James opens three instances of the **VLC media player** and accesses the appropriate URL for each playback method. In the URLs, `NGINX_server` is the IP address or hostname of his NGINX server:

- RTMP – `rtmp://NGINX_server/live/bbb`
- HLS – `http://NGINX_server/live/bbb.m3u8`
- DASH – `http://NGINX_server/live/bbb.mpd`

Improving the User Experience with NGINX Plus

If you're building a video library or course, you may need capabilities beyond what is possible with NGINX Open Source. NGINX Plus [includes extended features](#) to improve performance and the end-user experience with prebuilt modules. You can:

- Support higher scale by caching videos
- Provide video-on-demand (VOD) services
- Manage your streaming costs and capacity by limiting bandwidth

During the COVID-19 pandemic, NGINX is providing a free one-year license for up to five instances of NGINX Plus to organizations in the education, public government, and non-profit sectors (subject to review and approval). For details, see [Free Resources for Websites Impacted by COVID-19](#).

To incorporate the RTMP module into NGINX Plus, you load it dynamically. See the [NGINX Plus Admin Guide](#).

In this second demo, James shows how easy it is to set up NGINX Plus as a load balancer for three video servers, using Ansible and Terraform:



To access the files James uses in the second demo, see his [GitHub repo](#).

We're in This Together

If you're having trouble implementing video streaming, or any other use case, we're here to help! NGINX employees and the community are monitoring the [NGINX channel on Stack Overflow](#)¹ and responding to questions and requests as quickly as possible.

If you work for an organization on the frontlines of the pandemic and have advanced needs, you may qualify for the NGINX Plus licenses mentioned [above](#) as well as a higher tier of F5 DNS Load Balancer Cloud Service. See [Free Resources for Websites Impacted by COVID-19](#) for details.


Also check out that blog for a rundown of [easy ways to improve website performance](#) with free

¹Stack Overflow is a third-party website and is not affiliated with F5. Inc. F5 and its affiliates disclaim any liability for content (including general information and proposed solutions to questions) posted on Stack Overflow or any other third-party website.


 **NGINX Plus, video streaming, NGINX Open Source, COVID-19**


68 Comments



 Login ▾



LOG IN WITH



OR SIGN UP WITH DISQUS 










 3 • Share

Best Newest Oldest

K **Klarare**  
3 years ago edited

For me this config with hls works also with the free version of nginx. I was happy to find an example without ffmpeg. To display the stream you have to give the m3u8 playlist to your Javascript Videoplayer. Thanks to nginx and this example!



 7  0 Reply • Share ▾


brian botkiller  
2 years ago

It would be really good if you'd do things in this tutorial such as:



1. Point out that this seems to be more geared towards nginx plus and not nginx
2. Explain your config file additions and where the config file resides by default, and how to add these. You just sort of trail off in this section, both in the video and in the documentation, and it just leaves the user hanging. How are we to know where to add these bits about RTMP and such to the config file? Where is the config file? It takes a lot of digging to work out where it is, and you could have simply dropped into the tutorial, "By default, the nginx config file resides at blank blank", instead of just expecting that we know where it is?
3. Explain why "location" is important, specifically speaking to the fact that this denotes the stream itself. You also state to stream to /tv/tv2 using OBS, but no where do you define "tv2" as a location -- you only define "tv", so the part about OBS simply doesn't work.



This tutorial was alright but it has some serious holes in it.

 5  1 Reply • Share ▾



T **Tormy Van Cool**  
3 years ago





not possible to avoid KEY on OBS. It refuses to run into streaming

 2  0 Reply • Share ▾



A **Alexandr Borisovich**  
3 years ago edited



Latency 15s? It is useless for live streaming where you must get responses from auditory.

 2  0 Reply • Share ▾



N **NonLinear**    
a year ago

15 sec is a good scenario. I have 45 sec!!! It is useless.

 1  0 Reply • Share ▾

N **NonLinear**  
a year ago edited



I have implemented the recommended steps as have been described to have live streaming from nginx web server but at the end I listen only the audio but not webcam. The instructions are too simple to make a mistake. I set up an Ubuntu 20.04, install nginx with rtmp module, no ufw (firewall), configure the nginx.conf file as it needs etc... I use OBS to capture my audio and webcam and send it as RTMP streaming to nginx. No error in the nginx log. As player to play the RTMP stream I use the VLC. I tested the OBS to send RTMP to YouTube and everything ok. Also when I change the device capture in OBS from webcam to desktop or to stream a media file then it works!!! Why only the webcam doesn't show when I use nginx? Thanks

 1  0 Reply • Share ▾

Tony Mauro    
a year ago

I apologize, but we're not able to offer troubleshooting help in the comments section. I suggest sending your question to the NGINX mailing list, where fellow community members might be able to help (see <https://mailman.nginx.org/m...> for subscription instructions).

 1  0 Reply • Share ▾

A **Ahmad Abuziad**  
a year ago

original article: <https://www.nginx.com/blog/...>

Notes:

- article code snippets has a small difference than the video
- if you followed the video but couldn't see the stream from vlc then

[try to open port 80 and 1935](#)