Create your own video streaming server with Linux

Set up a basic live streaming server on a Linux or BSD operating system.

By Aaron J. Prisk

January 8, 2019 | 17 Comments | 12 min read

315 readers like this.

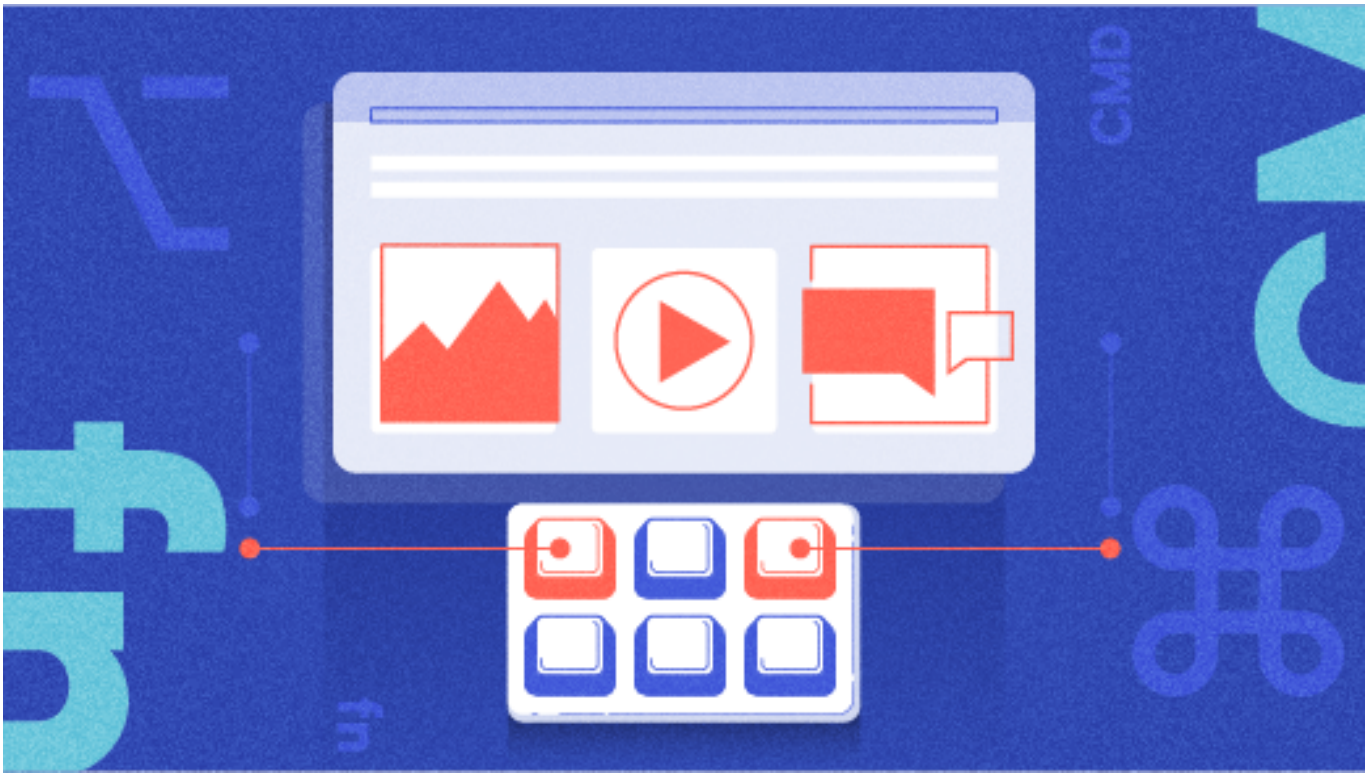


Image by: *Opensource.com*

Live video streaming is incredibly popular—and it's still growing. Platforms like Amazon's Twitch and Google's YouTube boast millions of users that stream and

consume countless hours of live and recorded media. These services are often free to use but require you to have an account and generally hold your content behind advertisements. Some people don't need their videos to be available to the masses or just want more control over their content. Thankfully, with the power of open source software, anyone can set up a live streaming server.

Getting started

In this tutorial, I'll explain how to set up a basic live streaming server with a Linux or BSD operating system.

This leads to the inevitable question of system requirements. These can vary, as there are a lot of variables involved with live streaming, such as:

Stream quality: Do you want to stream in high definition or will standard definition fit your needs?

Viewership: How many viewers are you expecting for your videos?

Storage: Do you plan on keeping saved copies of your video stream?

Access: Will your stream be private or open to the world?

More Linux resources

Linux commands cheat sheet

Advanced Linux commands cheat sheet

Free online course: RHEL Technical Overview

Linux networking cheat sheet

SELinux cheat sheet

Linux common commands cheat sheet

What are Linux containers?

Our latest Linux articles

There are no set rules when it comes to system requirements, so I recommend you experiment and find what works best for your needs. I installed my server on a virtual machine with 4GB RAM, a 20GB hard drive, and a single Intel i7 processor core.

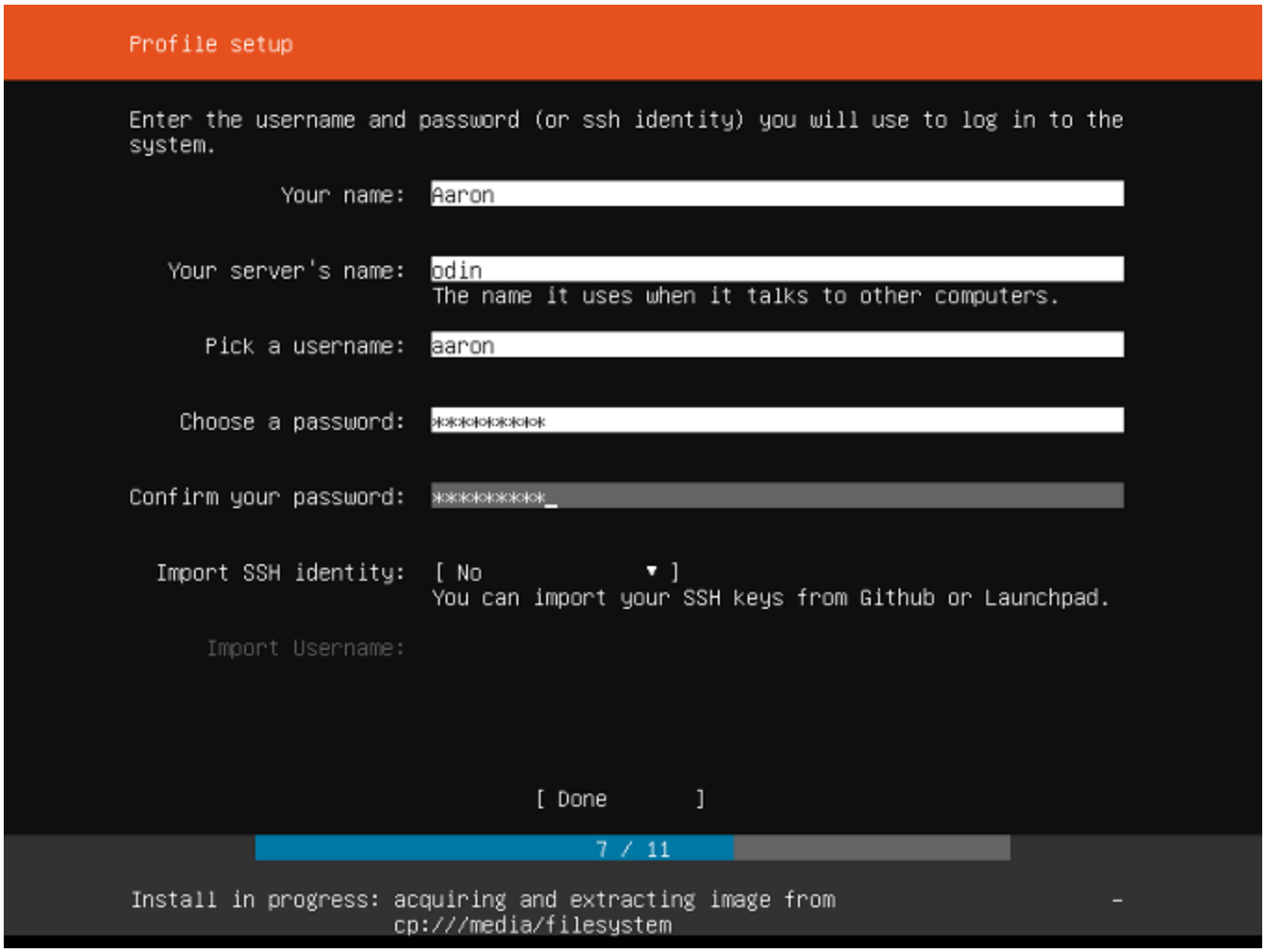
This project uses the Real-Time Messaging Protocol (RTMP) to handle audio and video streaming. There are other protocols available, but I chose RTMP because it has broad support. As open standards like WebRTC become more compatible, I would recommend that route.

It's also very important to know that "live" doesn't always mean instant. A video stream must be encoded, transferred, buffered, and displayed, which often adds delays. The delay can be shortened or lengthened depending on the type of stream you're creating and its attributes.

Setting up a Linux server

You can use many different distributions of Linux, but I prefer Ubuntu, so I downloaded the Ubuntu Server edition for my operating system. If you prefer your server to have a graphical user interface (GUI), feel free to use Ubuntu Desktop or one of its many flavors. Then, I fired up the Ubuntu installer on my computer or virtual machine and chose the settings that best matched my environment. Below are the steps I took.

Note: Because this is a server, you'll probably want to set some static network settings.



After the installer finishes and your system reboots, you'll be greeted with a lovely new Ubuntu system. As with any newly installed operating system, install any updates that are available:

```
sudo apt update
sudo apt upgrade
```

This streaming server will use the very powerful and versatile Nginx web server, so you'll need to install it:

```
sudo apt install nginx
```

Then you'll need to get the RTMP module so Nginx can handle your media stream:

```
sudo add-apt-repository universe
sudo apt install libnginx-mod-rtmp
```

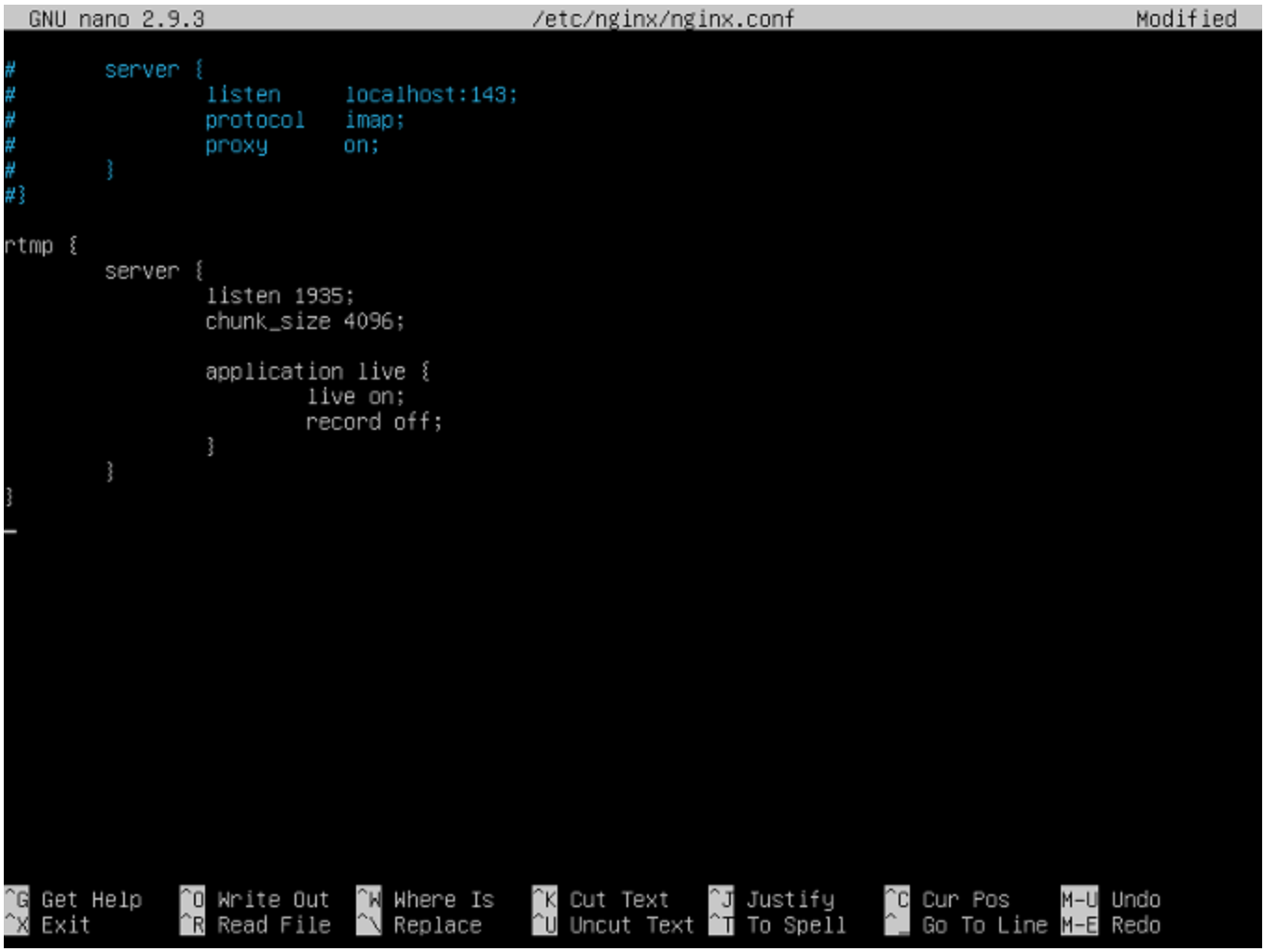
Adjust your web server's configuration so it can accept and deliver your media stream.

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

Scroll to the bottom of the configuration file and add the following code:

```
rtmp {
        server {
                listen 1935;
                chunk_size 4096;

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



Save the config. Because I'm a heretic, I use Nano for editing configuration files. In Nano, you can save your config by pressing **Ctrl+X**, **Y**, and then **Enter.**

This is a very minimal config that will create a working streaming server. You'll add to this config later, but this is a great starting point.

However, before you can begin your first stream, you'll need to restart Nginx with its new configuration:

```
sudo systemctl restart nginx
```

Setting up a BSD server

If you're of the "beastie" persuasion, getting a streaming server up and running is also devilishly easy.

Head on over to the FreeBSD website and download the latest release. Fire up the FreeBSD installer on your computer or virtual machine and go through the initial steps and choose settings that best match your environment. Since this is a server, you'll likely want to set some static network settings.

After the installer finishes and your system reboots, you should have a shiny new FreeBSD system. Like any other freshly installed system, you'll likely want to get everything updated (from this step forward, make sure you're logged in as root):

```
pkg update
pkg upgrade
```

I install Nano for editing configuration files:

```
pkg install nano
```

This streaming server will use the very powerful and versatile Nginx web server. You can build Nginx using the excellent *ports* system that FreeBSD boasts.

First, update your ports tree:

```
portsnap fetch
portsnap extract
```

Browse to the Nginx ports directory:

```
cd /usr/ports/www/nginx
```

And begin building Nginx by running:

```
make install
```

You'll see a screen asking what modules to include in your Nginx build. For this project, you'll need to add the RTMP module. Scroll down until the RTMP module is

selected and press **Space**. Then Press **Enter** to proceed with the rest of the build and installation.

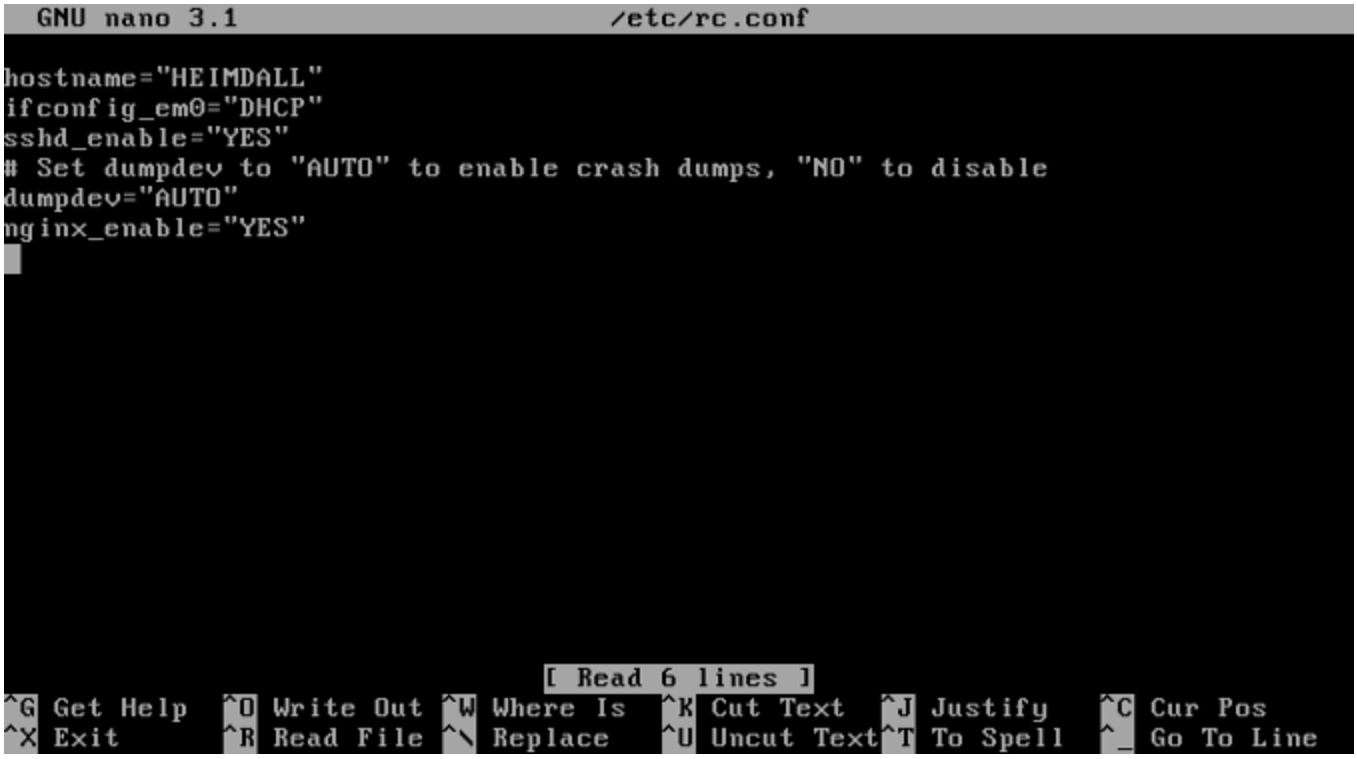
Once Nginx has finished installing, it's time to configure it for streaming purposes.

First, add an entry into **/etc/rc.conf** to ensure the Nginx server starts when your system boots:

```
nano /etc/rc.conf
```

Add this text to the file:

```
nginx_enable="YES"
```



Next, create a webroot directory from where Nginx will serve its content. I call mine **stream**:

```
cd /usr/local/www/
mkdir stream
chmod -R 755 stream/
```

Now that you have created your stream directory, configure Nginx by editing its configuration file:

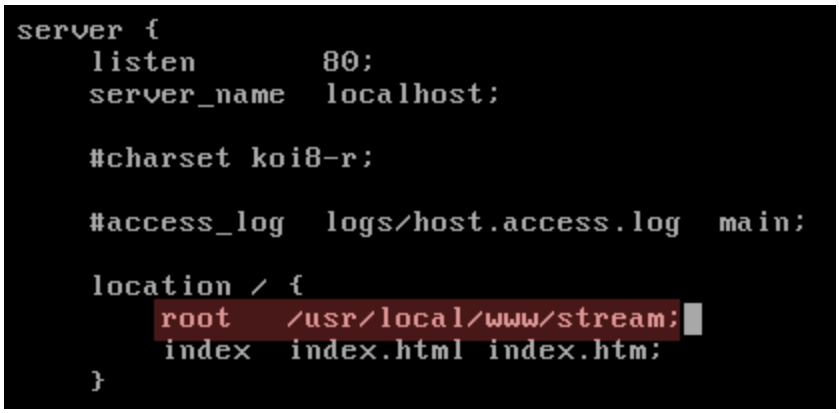
```
nano /usr/local/etc/nginx/nginx.conf
```

Load your streaming modules at the top of the file:

```
load_module /usr/local/libexec/nginx/ngx_stream_module.so;
load_module /usr/local/libexec/nginx/ngx_rtmp_module.so;
```

Under the **Server** section, change the webroot location to match the one you created earlier:

```
Location / {
root /usr/local/www/stream
}
```



And finally, add your RTMP settings so Nginx will know how to handle your media streams:

```
rtmp {
        server {
                listen 1935;
                chunk_size 4096;

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

Save the config. In Nano, you can do this by pressing **Ctrl+X**, **Y**, and then **Enter.**

As you can see, this is a very minimal config that will create a working streaming server. Later, you'll add to this config, but this will provide you with a great starting point.

However, before you can begin your first stream, you'll need to restart Nginx with its new config:

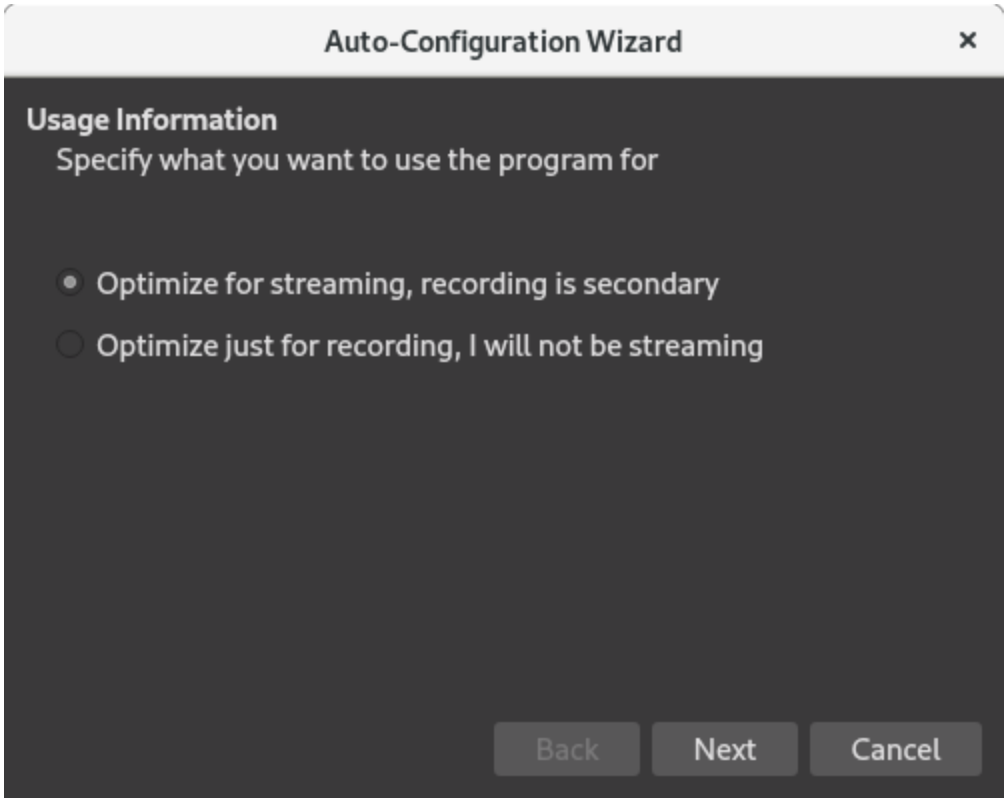
```
service nginx restart
```

Set up your streaming software

Broadcasting with OBS

Now that your server is ready to accept your video streams, it's time to set up your streaming software. This tutorial uses the powerful and open source Open Broadcast Studio (OBS).

Head over to the OBS website and find the build for your operating system and install it. Once OBS launches, you should see a first-time-run wizard that will help you configure OBS with the settings that best fit your hardware.



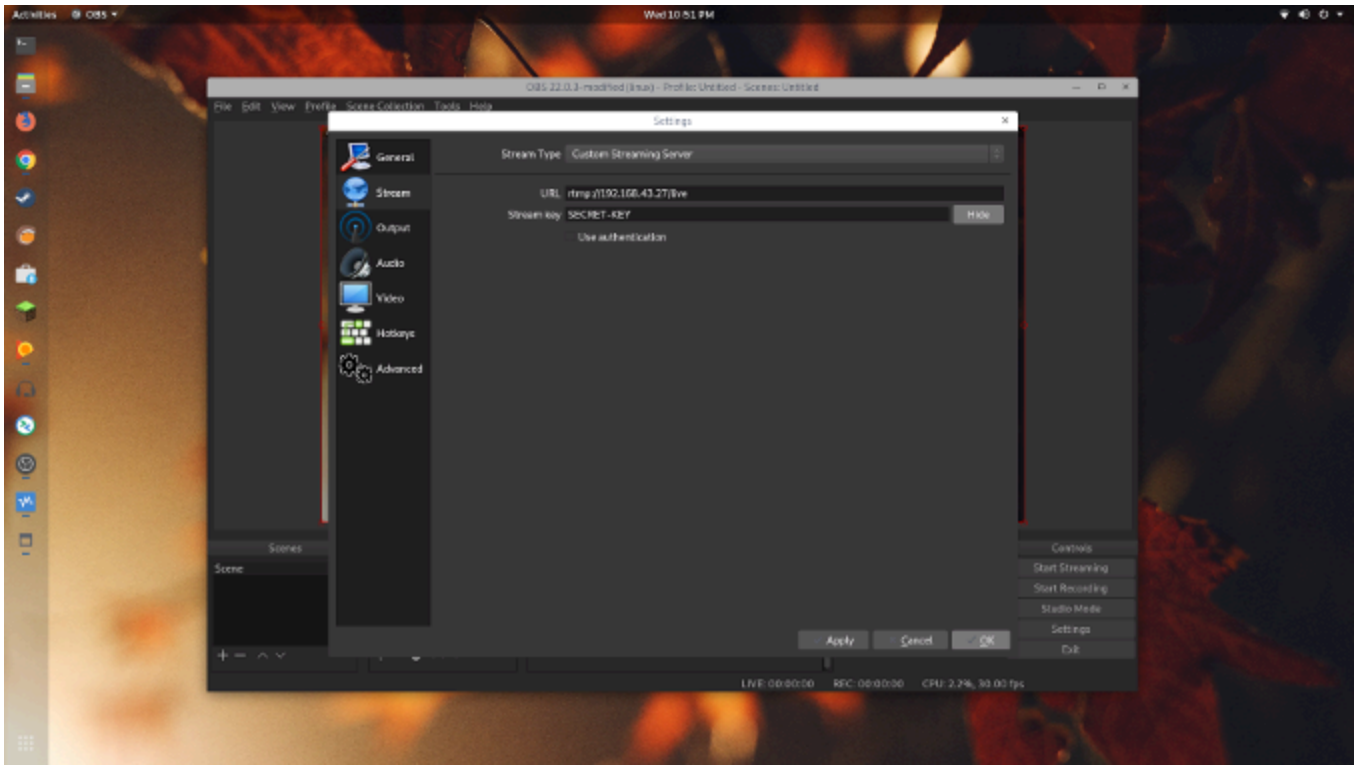
OBS isn't capturing anything because you haven't supplied it with a source. For this tutorial, you'll just capture your desktop for the stream. Simply click the **+** button under **Source**, choose **Screen Capture**, and select which desktop you want to capture.

Click OK, and you should see OBS mirroring your desktop.

Now it's time to send your newly configured video stream to your server. In OBS, click **File** > **Settings**. Click on the **Stream** section, and set **Stream Type** to **Custom Streaming Server**.

In the URL box, enter the prefix **rtmp://** followed the IP address of your streaming server followed by **/live**. For example, **rtmp://IP-ADDRESS/live**.

Next, you'll probably want to enter a Stream key—a special identifier required to view your stream. Enter whatever key you want (and can remember) in the **Stream key** box.



Click **Apply** and then **OK**.

Now that OBS is configured to send your stream to your server, you can start your first stream. Click **Start Streaming**.

If everything worked, you should see the button change to **Stop Streaming** and some bandwidth metrics will appear at the bottom of OBS.

If you receive an error, double-check Stream Settings in OBS for misspellings. If everything looks good, there could be another issue preventing it from working.

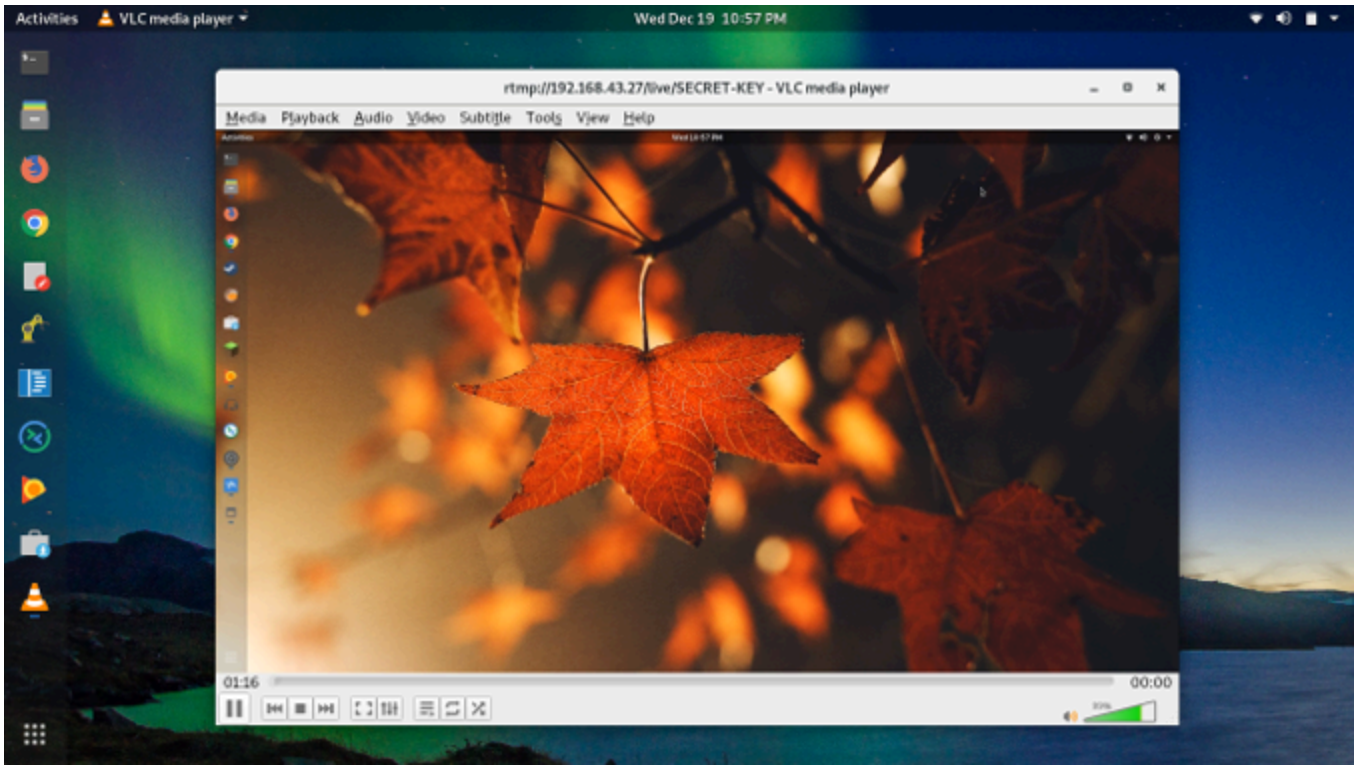
Viewing your stream

A live video isn't much good if no one is watching it, so be your first viewer!

There are a multitude of open source media players that support RTMP, but the most well-known is probably VLC media player.

After you install and launch VLC, open your stream by clicking on **Media** > **Open Network Stream**. Enter the path to your stream, adding the Stream Key you set up in OBS, then click **Play**. For example, **rtmp://IP-ADDRESS/live/SECRET-KEY**.

You should now be viewing your very own live video stream!



Where to go next?

This is a very simple setup that will get you off the ground. Here are two other features you likely will want to use.

Limit access: The next step you might want to take is to limit access to your server, as the default setup allows anyone to stream to and from the server. There are a variety of ways to set this up, such as an operating system firewall, .htaccess file, or even using the built-in access controls in the RTMP module.

Record streams: This simple Nginx configuration will only stream and won't save your videos, but this is easy to add. In the Nginx config, under the RTMP

section, set up the recording options and the location where you want to save your videos. Make sure the path you set exists and Nginx is able to write to it.

```
application live {
            live on;
            record all;
            record_path /var/www/html/recordings;
            record_unique on;
}
```

The world of live streaming is constantly evolving, and if you're interested in more advanced uses, there are lots of other great resources you can find floating around the internet. Good luck and happy streaming!

Tags: LINUX VIDEO EDITING

Aaron J. Prisk

School IT Director - Open Source Evangelist - Technology Enthusiast - Husband - Dad

More about me

17 Comments

These comments are closed.

Drew Kwashnak | January 9, 2019

No readers like this yet.

Great walk-through on setting it up yourself.

Aaron J. Prisk | January 9, 2019

No readers like this yet.

Thanks Drew! Glad you like it!

Stephen Melheim | January 11, 2019

No readers like this yet.

Sorry if this is an obvious question. But this is my first time setting up a media server. I'm using VirtualBox for the time to test the service. But where are you creating the application "live"? Also if you have something like a HDD that has music, videos, pictures, how is this going to connect to that?

Aaron J. Prisk | January 16, 2019

No readers like this yet.

Hey Stephen,
Think of "live" as just a container where your stream will be located. You could change that to "vid" or "app" and it would just change the URL that you'd have to point to.
EXP: rtmp://server/app/STREAM-KEY

The content that you want to show will be on the computer running OBS. In this setup, the server is just acting like a big repeater for the video that OBS is sending it.

Hopefully this helps clarify a little bit.

Himesh | January 17, 2019

No readers like this yet.

Great guide Aaron,
Getting a bit out of topic, but could you point me to an apt resource/video/guide to remotely access a VM.
I have created one on Google Cloud but am unable to access the same as on clicking the SSH, the command line opens.
Thanks a ton!

Aaron J. Prisk | January 17, 2019

No readers like this yet.

Hey Himesh,
I'm not super familiar with creating VMs on Google's platform. Being it's a remote server, you'll likely have to use SSH in order to connect and manage it.

Can you elaborate on what you're trying to do?

heap | January 27, 2019

No readers like this yet.

where is the firewall set up?

Aaron J. Prisk | April 2, 2019

No readers like this yet.

I didn't include it in my tutorial, but here's a good write-up for setting one up on Ubuntu:
https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewa...

And here's a pretty good write-up for using the PF firewall on FreeBSD: https://www.cyberciti.biz/faq/how-to-set-up-a-firewall-with-pf-on-freeb...

Alexander Brown | February 27, 2019

No readers like this yet.

How would you get this server to fetch ipcam footage and and allow to stream this.
i have a front door which has an ipcam, it only allows 1 viewer at a time, i want to see it from anywhere using this linux server and allow more than one person to stream simultneously

Vincent Stamos | March 5, 2019

No readers like this yet.

I am not sure how ipcam works, but if you view the cam through an ip address, you will just need to use the browser capture in the OBS software.

Aaron J. Prisk | April 2, 2019

No readers like this yet.

On OBS, you can add another RTSP stream as a Media Source by clicking the + under SOURCES > Media Source > Create New > Uncheck Local File > Under Input, paste the full RSTP stream of your ipcam.

I do something similar to do split screen game streaming where my friend and I's streams are shown side by side.

Ant Media Server | March 17, 2019

No readers like this yet.

Hi Aaron,
Thanks for the details. Live Streaming is so basic in Ant Media Server. You can review Ant Media Server. Ant Media Server is also Open Source Media Server. Ant Media Server Github Page: https://github.com/ant-media/Ant-Media-Server Ant Media Server Google Group: https://groups.google.com/forum/m/#!forum/ant-media-server Also review website: https://antmedia.io Best Wishes

Aaron J. Prisk | April 2, 2019
No readers like this yet.

Thanks for the reply! Ant Media Server is awesome and I definitely recommend it to people looking for a simple, turn key setup.

Alex P | March 18, 2019
No readers like this yet.

Thanks for the walk through

Does nginx provide the ability to timeshift ie rewind by x seconds when you press the left arrow? If not, can you recommend a way to do this.
Thanks

Aaron J. Prisk | April 2, 2019
No readers like this yet.

Not by itself as far as I know. You'd need some kind of player capable of caching the stream as it plays.

Rahul Dhiman | March 26, 2019
No readers like this yet.

Hey Araon thanks for writing this article..I wanted to know could i be able to access the files stored on my SERVER from a different pc(Client) and both the systems are connected via lan..

Aaron J. Prisk | April 2, 2019

No readers like this yet.

Sure can. First, you'd want to setup SSH on your streaming server then grant permissions to a user that you'd like to have access to that recordings directory. Once that's all set up, you can SSH into it and access those files.

If you're client is a Windows box, you could use WinSCP.

Related Content

What's new in GNOME 44?

5 reasons virtual machines still matter

Remove the background from an image with this Linux command

This work is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.

ABOUT THIS SITE

The opinions expressed on this website are those of each author, not of the author's employer or of Red Hat.

Opensource.com aspires to publish all content under a **Creative Commons license** but may not be able to do so in all cases. You are responsible for ensuring that you have the necessary permission to reuse any work on this site. Red Hat and the Red Hat logo are trademarks of Red Hat, Inc., registered in the United States and other countries.

A note on advertising: Opensource.com does not sell advertising on the site or in any of its newsletters.

Copyright ©2025 Red Hat, Inc.

Privacy Policy

Terms of use

Cookie preferences