Custom Streaming Setup

Amolith

2020-04-26T20:24:38-04:00

Contents

The other day, I decided that I wanted to start streaming. I'll definitely be playing some games but I might also stream some other things like me playing music. We'll see where that goes. In any case, I don't like relying on third parties for things and didn't want to use Twitch so I started figuring out how to build my own open source and privacy-friendly "platform" (which is really just a page.)

The search for a platform

Before settling on my own custom thing, I did some digging into ready-made platforms I could just throw on one of my servers and run. Two of the ones I found were OpenStreamingPlatform and Restreamer. The latter isn't exactly what I was looking for but it could have worked quite well. The former, at first glance, was absolutely *perfect*. On a functional level, it still is. However, take a look at the installation guide.

`<rant>`

Steps 3 and 7 are unnecessary unless you feel like manually compiling your web server; it's already available in the Debian repos and, by extension, Ubuntu's. It's even been backported to Stretch. In step 4, he has `sed -i 's/appendfsync everysec/appendfsync no/'`. Like so many application developers, he's assuming that this is the only project that will be installed on the system. If someone is already using redis in production and they have a different value there, that command will fail. In step 9, the commands are copying the SystemD service files to `/lib/systemd/` but this is where the package manager, `apt`, stores its services. When you have your own that you're writing or copying from somewhere else, best practise is to put them in `/etc/systemd/system`. In addition, all of this is scripted for the "standard" install. Yes, you're always supposed to review scripts before running them but who really does that? When I see a project whose only supported installation method is a script, I nope right on out of there for exactly this reason. I know how my system *is* set up and I know how I *want* it set up. I can't stand it when they assume they know what's best. Just tell me what you *recommend* and I'll make decisions from there.

`</rant>`

NGINX & RTMP

RTMP stands for Real-Time Messaging Protocol and facilitates streaming audio, video, and other data over the internet in real-time. The NGINX module mentioned above adds functionality to NGINX that allows it to handle RTMP streams and turn them into something a browser or media streaming client can use. Connecting directly via `rtmp://example.com/live/stream` is not very widely supported so protocols such as MPEG-DASH and HLS are used instead.

On Debian-based systems, adding RTMP functionality to NGINX is as simple as `apt install libnginx-mod-rtmp`. After that, you'll need to add some things to your `nginx.conf` and whatever host file you're using for your website.

```
rtmp {
  server {
    listen 1935;
    application live {
      deny publish all;
      allow publish 127.0.0.1;
      live on;
      interleave on;
      hls on;
      hls_path /tmp/hls;
      hls_fragment 15s;
      dash on;
      dash_path /tmp/dash;
      dash_fragment 15s;
    }
  }
}
```

`1935` is the default RTMP port. `deny publish all` means you are denying *anyone* from publishing a stream (that includes you. `allow publish 127.0.0.1` allows *local* connections to publish content. I'm using this as a form of authentication—before streaming anything, I have to tunnel my connection to my server via SSH or a VPN. At the moment, I'm using SSH:

```
ssh -L 1935:localhost:1935 user@example.com
```

The other options are just the basics needed to get DASH and HLS to work. The only other thing to do is use NGINX as a reverse proxy (sort of) to serve the streams. Add this to your site's virtual host.

```
location /dash {
  root /tmp;
}
location /hls {
  root /tmp;
}
```

That's it! Now you'll need to test your stream and verify that it actually works.

```
ffmpeg -re -i video.mp4 -vcodec copy -loop -1 -c:a aac -b:a 160k -ar
    44100 -strict -2 -f flv rtmp://example.com/live/stream
```

This command has FFmpeg play the video and stream it to the server. You should then be able to open the stream in something like VLC or MPV and watch it from anywhere.

```
mpv https://example.com/dash/stream.mpd
```

However, I also wanted to embed it in a website and this is where it gets a little unstable.

Browser playback

`dash.js` is currently one of the best ways to play a live stream in a browser plus it's pretty easy to work with. The code can be found on GitHub. Using the setup with NGINX I detailed above, this should work perfectly fine out of the box.

```
<div>
    <video id="videoPlayer" poster="/assets/jpgs/stream.jpg"
        controls></video>
</div>
<script src="/assets/js/dash.all.min.js"></script>
<script>
(function(){
    var url = "/dash/stream.mpd";
    var player = dashjs.MediaPlayer().create();
    player.initialize(document.querySelector("#videoPlayer"), url,
        true);
})();
```

```
</script>
```

Web chat

The last thing every stream needs is something for web chat. I tried a few different solutions and had mixed results. The first was KiwiIRC but the iframe wouldn't even finish loading because it connected to so many third parties with a lot of tracking. It functions very well and I might set it up on my own site eventually but it was a bit much to go through at the time. As an intermediate solution, I embedded my instance of The Lounge, a fully-functional web-based IRC client. This loaded perfectly right out of the box but it wasn't quite what I wanted; there were *too* many options and the friends of mine who tested it got frustrated because some of the essential UI elements were hidden due to the small viewport. It's just not quite suitable for embedded webchat.

Finally, I landed on qwebirc and it was pretty much *exactly* what I wanted. When the iframe loads, you're prompted to enter a nick, you click connect, wait a minute, and done! My one complaint is that the theme is very bright but I'll work on that later on. It's good enough for now :wink:

EDIT: Since the time of writing, I have switched to hosting KiwiIRC on Secluded.Site so all of the trackers and third parties aren't in use. My configs are below but I recommend going through the wiki and making your own decisions.

`/etc/kiwiirc/config.conf`

```
logLevel = 3
identd = false
gateway_name = "webircgateway"
secret = "changeme"

[verify]
recaptcha_secret = ""
recaptcha_key = ""

[clients]
username = "%i"
realname = "KiwiIRC on secluded.site"

[server.1]
bind = "0.0.0.0"
port = 7264

[fileserving]
enabled = true
webroot = /usr/share/kiwiirc/

```

```
[transports]
websocket
sockjs
kiwiirc

[reverse_proxies]
127.0.0.0/8
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
"::1/128"
"fd00::/8"

[upstream.1]
hostname = "irc.nixnet.services"
port = 6697
tls = true
timeout = 5
throttle = 2
webirc = ""
```

/etc/kiwiirc/client.json

```
{
    "windowTitle": "Secluded.Site Chat",
    "startupScreen": "welcome",
    "kiwiServer": "/webirc/kiwiirc/",
    "restricted": true,
    "hideAdvanced": true,
    "showAutoComplete": true,
    "showSendButton": true,
    "sidebarDefault": "nicklist",
    "theme": "dark",
    "themes": [
        { "name": "Default", "url": "static/themes/default" },
        { "name": "Dark", "url": "static/themes/dark" },
        { "name": "Coffee", "url": "static/themes/coffee" },
        { "name": "GrayFox", "url": "static/themes/grayfox" },
        { "name": "Nightswatch", "url": "static/themes/nightswatch"
            },
        { "name": "Osprey", "url": "static/themes/osprey" },
        { "name": "Radioactive", "url": "static/themes/radioactive"
            },
        { "name": "Sky", "url": "static/themes/sky" }
    ],
  "buffers" : {
    "messageLayout": "compact",
    "show_timestamps": false,
    "show_hostnames": false,
    "show_joinparts": false,
    "show_topics": true,
    "show_nick_changes": true,
    "show_mode_changes": false,
    "traffic_as_activity": false,
    "coloured_nicklist": true,
    "colour_nicknames_in_messages": true,
    "block_pms": true,
    "show_emoticons": true,
    "extra_formatting": true,
    "mute_sound": false,
    "hide_message_counts": false,
    "show_realnames": false,
    "default_kick_reason": "Your behaviour is not conducive to this environment.",
    "shared_input": false,
    "show_message_info": true,
    "share_typing": true,
    "flash_title": "off",
    "nicklist_avatars": true,
    "show_link_previews": true,
    "inline_link_previews": true,
    "inline_link_auto_preview_whitelist": "secluded.site|nixnet.services",
    "channel": "#secluded"
  },
    "startupOptions" : {
        "server": "irc.nixnet.services",
        "port": 6697,
        "tls": true,
        "direct": false,
        "channel": "#secluded",
        "nick": "viewer?",
        "greetingText": "Welcome!",
        "infoBackground": "",
        "infoContent": ""
    }
}
```

Actually streaming

Once you're ready to start streaming content, I recommend using OBS Studio. If you're noticing issues with stream performance, play around with your output resolution and FPS—those are the biggest factors. To use OBS with NGINX, you'll need to go to `Settings`, `Stream`, and set `Server` to `rtmp://localhost/live/`. If you're using my configs as they are, the key will need to be `stream`. Literally every component requires specific paths so, unless you're careful, things will break and you'll spend hours trying figure it out like I did. Also don't forget that the connection *has* to be tunnelled if you want authentication as I mentioned above. If you don't have `localhost:1935` on your streaming machine tunnelled to port 1935 on your server, OBS is going to throw errors about not being able to connect.

Summary

I'm pretty proud of the set up I have now but it could still do with some improvements. For example, I plan to mess with the CSS and make both the video and chat panes *much* wider as well as side-by-side rather than on top of each other. Everything is crammed together and it's not a very good experience.

References

This post has pieces taken from a few other articles and sites that also deserve a mention as well as a read. NGINX actually has an official blog post on setting up RTMP streaming (though they compile NGINX from source as well) that was a *massive* help. I also found another post that is very similar to this one about HTML5 Live Streaming with MPEG-DASH. A good number of the parts are the same but I used the NGINX module in Debian repos and they used a fork of it with additional features. My NGINX setup was mostly from the NGINX blog post and the embedded stream was primarily from Inanity's. I figured out some of the components I could use for all of this from Drew DeVault.

This was posted as part of #100DaysToOffload, an awesome idea from Kev Quirk. If you want to participate, just write something every day for 100 days and post a link on social media with the hashtag!