published 8/21/2025

The First MoQ CDN: Cloudflare

🚨 It’s finally happening! 🚨

Cloudflare has **just announced** their Media over QUIC CDN! It’s an **official product**, and you can test MoQ on their *massive*, anycast network. Try it out, and convince your boss’ boss that the writing is on the wall.

If you’ve been living under a rock, MoQ is an **up-and-coming standard** for live media, aiming to supplant **WebRTC**, **HLS/DASH**, and even **RTMP/SRT** as the one to rule them all. And now Cloudflare wins the award for the first CDN offering!

Your prize is a blog post. You’re welcome mega-corp.

Also, *while you’re here*, some shameless self-promotion: I just soft-launched **hang.live**. Check it out if you want to see the cringe cool stuff you can do with MoQ.

What’s available now?

This is a **technical preview**, so it's both free and subject to change.

Cloudflare is hosting a public `relay.cloudflare.mediaoverquic.com` endpoint that you can abuse test. Connect using **my library**, **Mike's fork**, **Lorenzo's imquic**, **Meta's moxygen**, or any client that supports this limited subset of draft-07.

I'm biased so naturally I'm going to use **@kixelated/hang** (smash that star button). You can publish a live broadcast in the browser using the **web demo** or the **library**:

```
<script type="module">
    // Registers the <hang-publish> element.
    import "@kixelated/hang/publish/element";
</script>

<!-- You'll need to replace `name` with something unique/random. -->
<hang-publish url="https://relay.cloudflare.mediaoverquic.com" name="unique-name-
    <!-- It's optional to provide a video element to preview the outgoing media.
    <video style="max-width: 100%; height: auto; border-radius: 4px; margin: 0 au
</hang-publish>
```

There's a link to watch your live broadcast using the **web demo**, or again you can use the **library**:

```
<script type="module">
    // Registers the <hang-watch> element.
    import "@kixelated/hang/watch/element";
</script>

<!-- Use the same name as the broadcast you published. -->
<hang-watch url="https://relay.cloudflare.mediaoverquic.com" name="unique-name-ab
    <!-- It's optional to provide a canvas if you want audio only -->
    <canvas style="max-width: 100%; height: auto; border-radius: 4px; margin: 0 a
</hang-watch>
```

You might even notice **closed captions** because I've been experimenting with AI features (gotta get funding eventually 💰). They're generated *in the*

browser using **silero-vad** + **whisper** + **transformers.js** + **onnxruntime-web** + **WebGPU** and transmitted using MoQ of course. But that's a whole separate blog post; it's pretty cool.

NOTE: You don't have to use this **Web Component** API. **hang.live** uses the far more powerful Javascript API to do more complicated stuff like get access to individual video frames. There's a *super secret* section at the end of this blog if you LOVE sample code, but I'm not going to bore the rest of you.

There's also a 🦀 Rust 🦀 library **to import MP4**, **pipe media from ffmpeg**, and **publish/watch using gstreamer** so you can do more complicated media stuff without 🤮 Javascript 🤮. I wish I could spend more time on the Rust side but **WebSupport** is a big deal. We are no longer forced to use WebRTC, but that also means we need to build our own WebRTC in 🤮 Javascript 🤮. I can suffer and you can reap the rewards.

(and yes, **I'm aware that WASM exists**, but I ended up abandoning it)

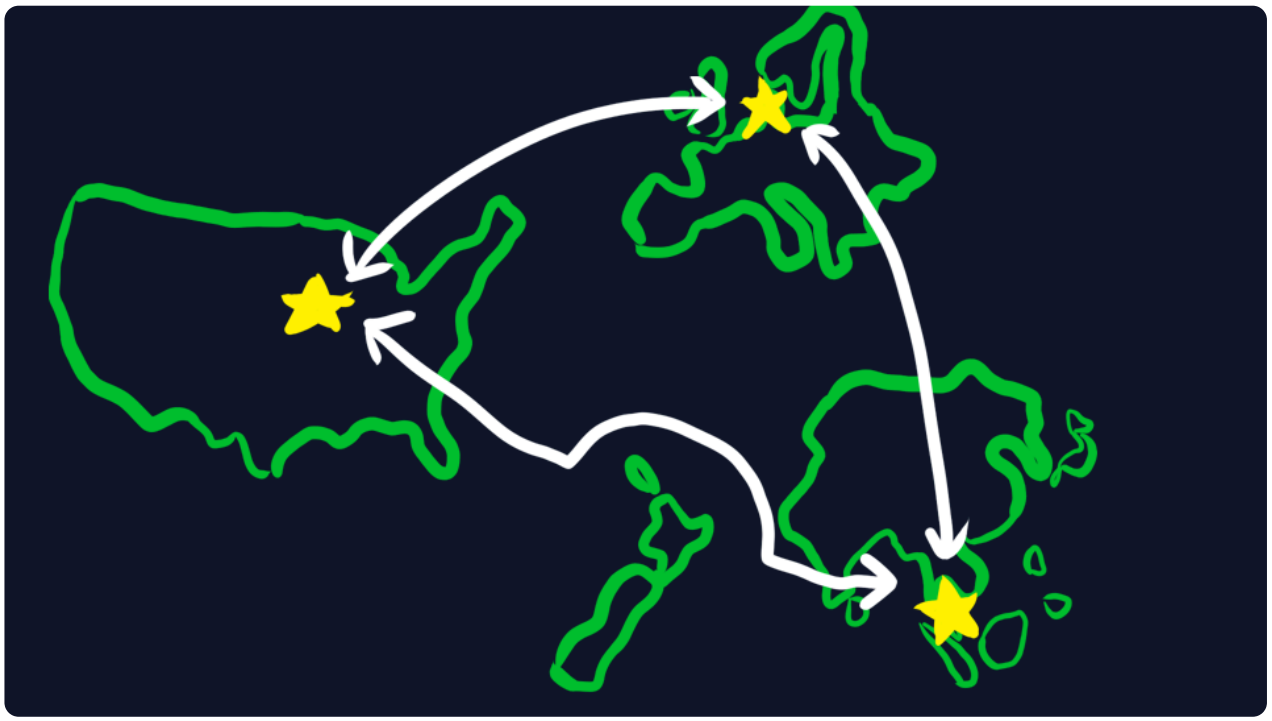
What's not available yet?

This is a **preview** release. Cloudflare is only supporting a *tiny* subset of an **old draft**, which is even smaller than **my tiny subset**. They're using a **fork** of my terrible code so bugs are guaranteed.

- **There's no authentication yet**: choose an unguessable name for each broadcast.
- **There's no ANNOUNCE support**: my **conferencing example** uses **ANNOUNCE** to discover when broadcasts start/stop, so that won't work.
- **There's no Safari support**: **It's coming eventually**.
- **Nothing has been optimized**: the user experience will improve over time.

If any of these are deal breakers, then you could always run your own **moq-relay** in the meantime. I've been adding new features and fixing a bunch of stuff *after* Cloudflare smashed that fork button. For example, authentication (via JWT) and a WebSocket fallback for Safari/TCP support.

There's even a **terraform module** that powers `relay.moq.dev`. You too can run your own "global" CDN with 3 nodes and pay GCP a boatload of money for the privilege. It's not *quite* as good as Cloudflare's network, currently available for free...



A "global" CDN according to me, an American. At least I didn't **forget New Zealand**.

Or host **moq-relay** yourself! It should even work on private networks provided you **wrestle with TLS certificates**. I'd also love to get MoQ running over **Iroh** for peer-to-peer action if anybody wants to help.

Why should you care?

As a great philosopher once said:

> *"Apathy is a tragedy and boredom is a crime. - **Bo Burnham**"*

This is a big deal. The biggest of deals. The HUGEST of deals.

I've been an **outspoken critic** of the MoQ standardization process. It's just really difficult to design a protocol, via a cross-company committee, before there's been any real world usage. It's been over 3 years since I fought

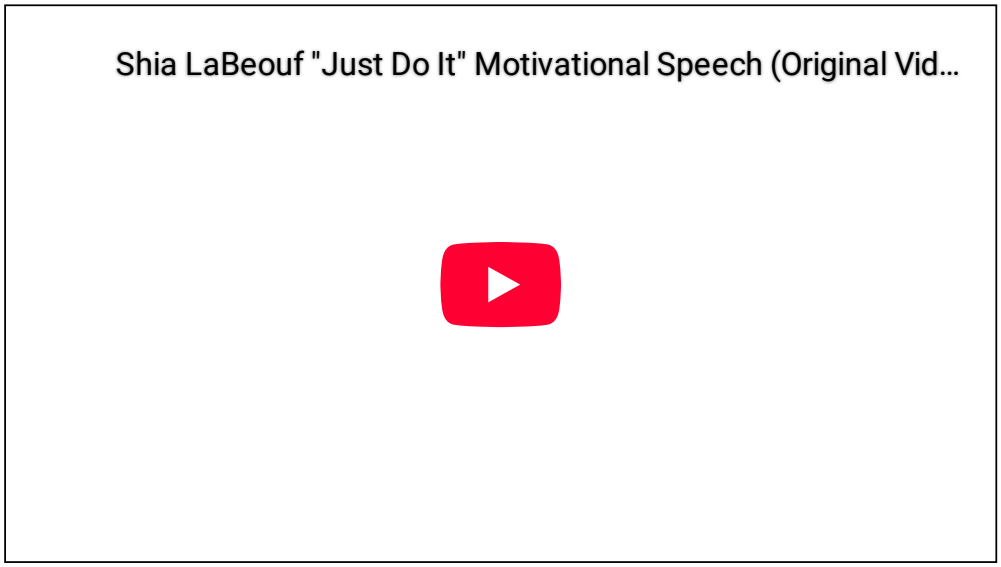
Amazon lawyers and published my **first MoQ draft**. It's going to be at least another 3 years before even the **base networking layer** becomes an RFC.

And that's by design! The best standards take a while. Look no further than QUIC, deployed by Google in 2012, started standardization in 2015, with the RFC released in 2021. And they had a boatload of production data to shape the specification. Meanwhile, we have only had a **Big Buck Bunny demo**, and I believe the standard has veered off course as a result.

Cloudflare has done something fantastic and said:

> *"fuck waiting for a RFC, let's release something"*

Okay they didn't say that, but this is **exactly** the mentality that MoQ needs right now. **Just build something**. **Just release something**. **Just do it**.



Holy shit I'm Shia LaBeouf.

Arguing in the **650+ issues** and **500+ PRs** can wait for another day. Tweaking the messaging encoding for the hundredth time can wait for another day. We're still going to make sure that MoQ gets standardized *eventually*, but it's more important to get *something* out there.

I'm looking at you: Google, Akamai, Fastly, etc. Take some code, run it on some spare servers, and start to learn what customers need *before* you design the protocol.

What's next?

A lot of stuff.

We're effectively trying to reimplement WebRTC / HLS / RTMP using relatively new Web APIs. Don't judge MoQ based on these initial offerings. We've got a **ton** of work to do. **Let's do it**.

Join the Discord. Somehow there's 900+ people in there. Ping me and I will do whatever I can to help. *Especially* if it means putting one more nail in the WebRTC coffin.

Written by **@kixelated**.

Javascript is an Abomination

Still reading?

You win some bonus documentation. Congrats! I knew you would win.

Here's an example of my reactive library in action. It powers **hang.live** so the API is subject to change and is probably already out of date. When in doubt, **consult the source code** like the hacker you are.

```
import { Watch } from "@kixelated/hang"

// Start downloading a broadcast.
const watch = new Watch.Broadcast({
    enabled: true,
```

```
    url: "https://relay.cloudflare.mediaoverquic.com",
    name: "unique-name-abc123",
    video: { enabled: true },
    reload: false, // required for Cloudflare's CDN
});

// You can toggle reactive properties.
watch.audio.enabled.set(true);

// There are helpers to convert my custom signals, like for React:
import react from "@kixelated/signals/react"
const audioInfo = react(watch.audio.info); // a JSON blob of track information

// You could use the built-in renderers.
const canvas = document.getElementById("canvas");
const audio = new Watch.AudioEmitter(watch.audio, { volume: 0.5 });
const video = new Watch.VideoRenderer(watch.video, { canvas });

// Or you can do it yourself, like this crude Vanilla JS example:
const dispose = watch.video.frame.subscribe((frame?: VideoFrame) => {
    if (!frame) return;

    // Render the frame to a canvas, or pass it to a ML model, or whatever.
    canvas.getContext("2d")?.drawImage(frame, 0, 0);

    // NOTE: You should use requestAnimationFrame instead, but I'm lazy.
});
```

There's even some *top-secret* features behind undocumented APIs. Like running an object detection model in browser and publishing the results as a MoQ track. Stay tuned for a blog post about that if I can figure out a better use-case than a cat cam. 🐈

```
// Publish a broadcast.
const publish = new Watch.Publish({
    enabled: true,
    url: "https://relay.cloudflare.mediaoverquic.com",
    name: "unique-name-abc123",
    device: "camera",
    video: {
```

```
            enabled: true,
            detection: {
                enabled: true,
            }
        },
    })
```

Also, for the record, Typescript is really nice. 🤮 Javascript 🤮 is still an abomination.