Piracy Deployment in the era of AI

Abstract

Modern digital piracy is an ever evolving industry and it has evolved from amateur server operations into highly sophisticated, decentralized, and automated networks. As copyright holders and Internet Service Providers (ISPs) increasingly rely on DNS and IP blocking to stop illegal streaming, the effectiveness of these measures are questionable in the era of AI. Modern piracy groups operate with high efficiency, utilizing containerization, continuous deployment, and instant communication channels like Discord or Telegram to bypass blocks in real-time. This study examines how piracy websites can be easily created, cloned and deployed, and why blocking at DNS/IP level is not the solution in the period of the AI boom.

Architecture

Modern piracy websites have shifted from hosting the websites and the content on one server to more sophisticated setups heavily utilizing automation, cloud native architectures that make taking them down nearly impossible.

Decoupled Infrastructure

Modern websites are split into many separate parts, each running on its own dedicated computer called a server. First, a user passes through a middleman computer called a proxy, which protects the site and easily swaps out for a backup proxy if it gets blocked (Cloudflare, etc). Next, the user sees the frontend, which is just the visual layout of the website. To actually display information, this visual frontend asks the backend, the behind-the-scenes brain of the site, to fetch the data. The backend then gathers all the necessary info from separate storage computers (called databases) and outside digital tools (called APIs), bringing it all together to load the final webpage for the user.

Containerization

Piracy developers package their website's frontend, user databases, and content scrapers into isolated Docker[1] containers. If a hosting provider shuts down a pirate server, the operators simply take their container image and deploy it on a new "bulletproof" host in a different jurisdiction within seconds.

Infrastructure as Code (IaC)

Using tools like Terraform[2] or automated scripts, pirate operators can automatically provision new servers, configure reverse proxies, and restore database backups with a single command.

[1] Docker is a tool that wraps software into a secure "box" so it runs perfectly on any computer.
[2] Terraform is a tool that lets developers write a blueprint for their digital infrastructure (like servers and databases) so it can be automatically built and managed.

The cloning of a website

To understand the operational mechanics of these platforms, an experimental approach involving the creation of a functional replica, or 'clone,' was undertaken. This exercise simulated the perspective of an individual with little to no knowledge on how such websites operate.

Core requirements

Domain name

The initial step involved conceptualizing a name and obtaining a domain for the experimental site. The operators of the warez websites buy domain names in bulk with unusual TLDs (Top Level Domains) such as “.sx”, “.to”, “.li”. They do so to have the ability to run as many mirrored sites as possible to prevent service downtime in case of domain seizures or takedowns. Alternatively the domain names can be rotated to further lower the risk of the seizures or takedowns.

An example could be cineby.sc and bitcine.tv both sites have identical visuals, excluding some color changes. The current state of cineby.sc and bitcine.tv is backed up using Wayback Machine.

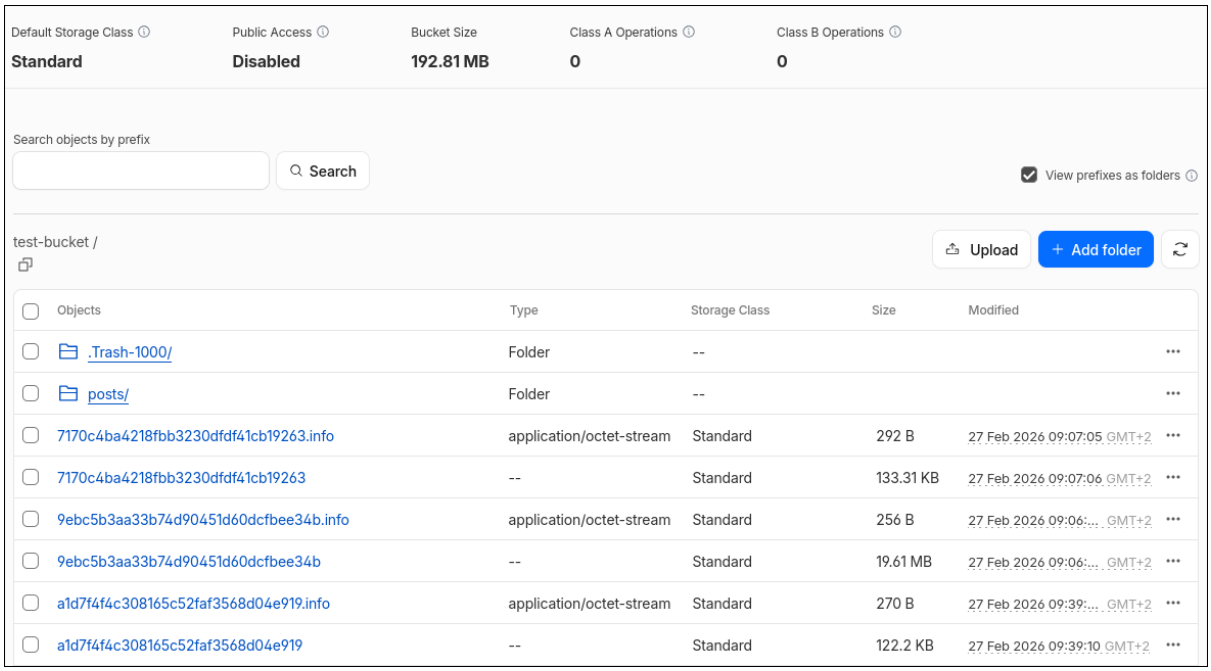
Infrastructure and hosting

The operators rely on complex, distributed infrastructure, legitimate hosts and CDNs are often abused or attacked. Operators typically rent servers and bandwidth across multiple countries (often choosing jurisdictions with lax enforcement or “bulletproof” hosting providers). This decentralization aims to eliminate single points of failure that could lead to service disruption.

For our experiment we decided to store all of the data on one primary server as we did not expect a lot of traffic and it is the best for the sake of simplicity. As previously mentioned, real websites tend to be heavily decentralized.

The most common setup is to have a user facing website acting as an aggregator of content stored on other file hosting services, these user facing websites then claim that they do not take responsibility for any of the embedded content.

The backend hosting infrastructure ranges from regular servers running services such as Nginx[3] for static serving of files to more involved configurations employing reverse proxies and S3-compatible storage solutions[4].



The above screenshot depicts the UI of Cloudflare's R2 Object Storage, a S3 compatible storage.

Using S3-compatible storage is great for operators because it uses a single, standard communication language that almost all modern cloud providers understand. Because everyone speaks this same language, if an operator gets banned by one storage company, they can easily move all their files to a new company without having to rewrite any website code—they just have to update their login passwords or basic setup settings. To stay safe, operators often keep identical copies of their files (called mirrors) saved across several different companies at the exact same time. On top of that, these storage services handle all the technical heavy lifting automatically, meaning they can scale up to handle a virtually infinite amount of user traffic without crashing. While it is technically possible to let users download files directly from these storages over the internet without going through a protective middleman server, this "read-only" setup is very rare, and we have not actually seen any real websites use it yet.

For the user facing websites, operators have several architectural choices, ranging from fully static sites to dynamic web applications.

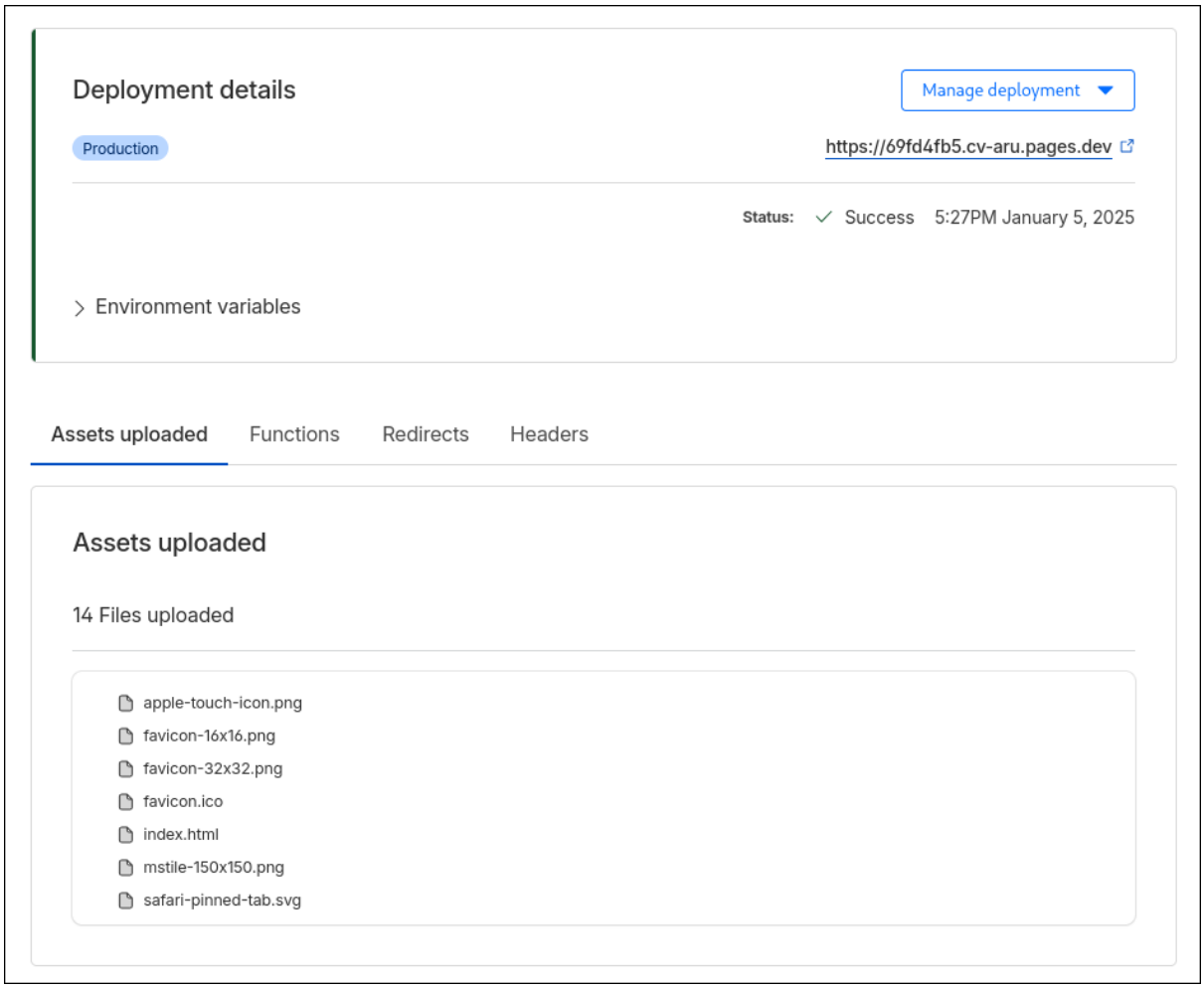
The fully static solution is somewhat limited as it means authoring the website’s content ahead of time and every user is going to receive the same exact content, this means that

[3] Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.

[4] The S3 protocol refers to the application programming interface (API) used by Amazon Simple Storage Service (S3) for managing and accessing data stored in the cloud. It allows developers to perform operations like uploading, downloading, and managing data objects in a scalable and efficient manner.

there is no dynamic server part which could be used to obfuscate for example the data sources. This model is suitable for simpler use cases, such as sharing a single live stream, where server-side rendering benefits are not critical. A key advantage is the potential to leverage free serverless platforms, which offer scalability comparable to managed S3-compatible storage.

An example of free static hosting providers might be pages.cloudflare.com. This service lets users deploy their website into many locations at no costs and almost no usage limits.



In the above screenshot the deployment of the website was done manually by dragging the website files into the interface, in a few seconds we were provided with a website publicly available and we could start sharing the auto generated website address with potential consumers of our pirated content.

In practice, however, these websites are typically more dynamic. From what we managed to gather PHP[5] is a widely used server side language. The specific programming language is often secondary, as the primary function of the web server is to retrieve data from a data source, such as a database or possibly a third party service for example providing various information about the content. Many of these websites have some sort of system for

[5] PHP is a general-purpose scripting language geared towards web development.

searching and categorizing the content and this can be written in almost any language. For our clone we decided to use JavaScript[6] with the Bun runtime[7] as it allowed us to take some shortcuts thanks to some nice built-in features.
An example of a website using PHP might be bombuj.si where the URL of the landing page ends in a file called *uvod.php*. This website even has categories, search and some form of user accounts features.

The current state of bombuj.si can be found here.

[6] JavaScript, often abbreviated as JS, is a programming language and core technology of the World Wide Web, alongside HTML and CSS.
[7] Bun is a JavaScript runtime, package manager, test runner and bundler built from scratch using the Zig programming language.

Bypassing the DNS system

Traditionally acquiring a domain name is considered a mandatory step for launching a website. It acts as a human-readable address, however bypassing this step entirely is a highly effective strategy to avoid detections, reduce overhead and circumvent DNS blocking.

Direct IP Indexing by Search Engines

Every server connected to the internet is assigned a unique string of numbers known as an **Internet Protocol (IP) address** (for example, `192.0.2.1`). If a web server is configured to accept direct connections, any internet user can access the site simply by typing that string of numbers directly into their browser's address bar.

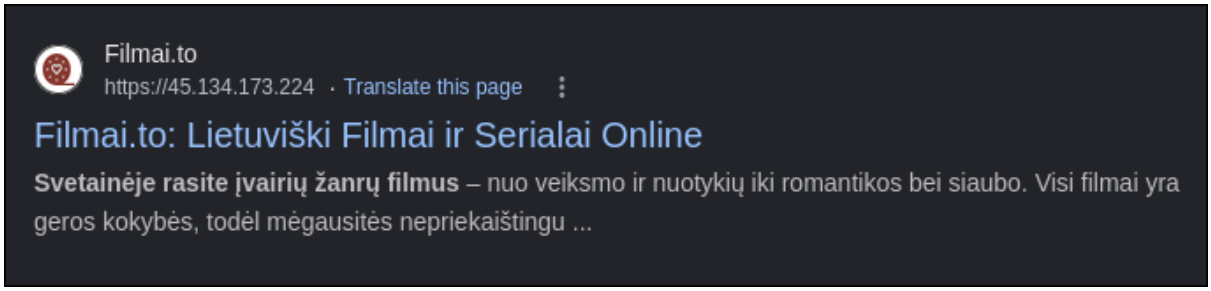
Critically, automated web crawlers operated by major search engines (such as Googlebot) do not rely solely on human-readable domain names to discover content. These search crawlers continuously scan the internet by randomly or systematically probing blocks of IP addresses.

If a crawler encounters an active web server hosting content at a raw IP address, it treats it like any other website. Users can discover the platform directly via a standard search engine query, even though the result appears as a string of numbers rather than a brand name.

Bookmarking

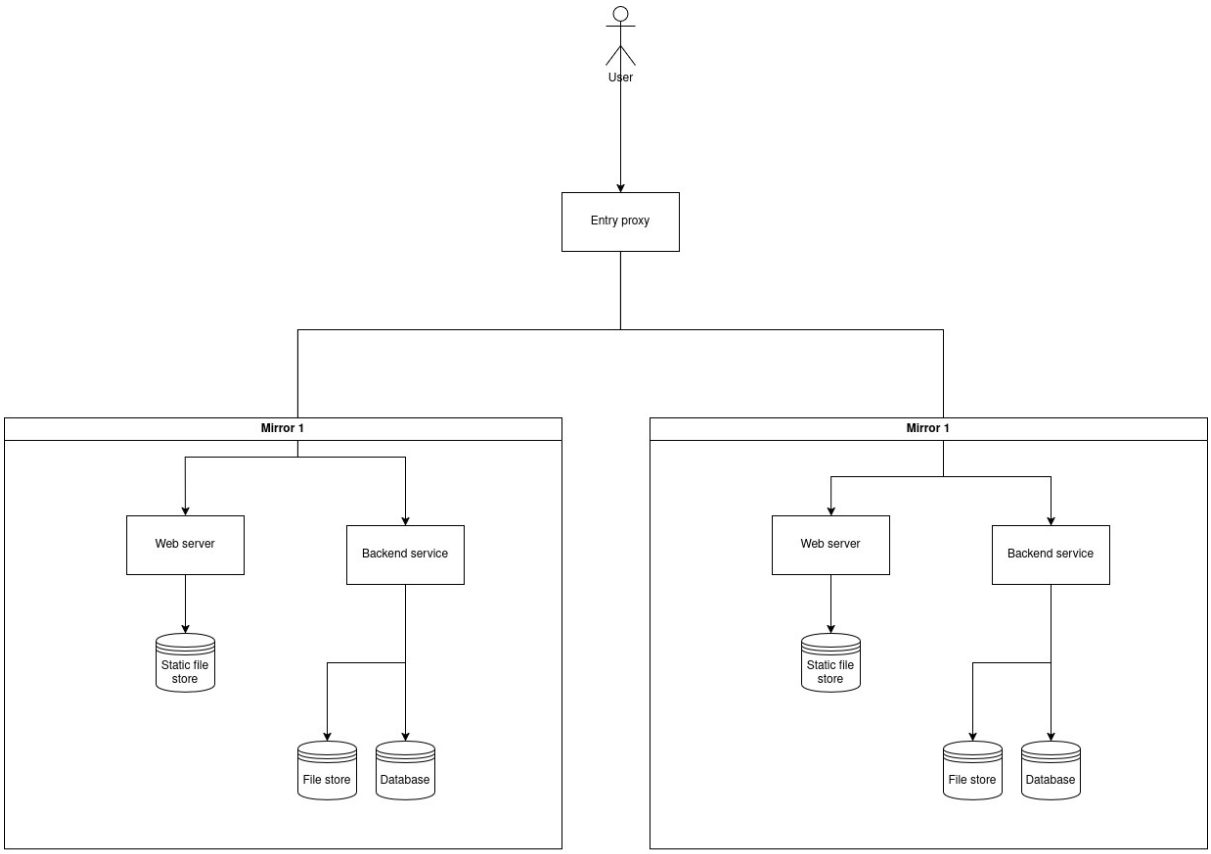
While a string of numbers is inherently harder for a human to memorize than a word, this minor friction does not meaningfully deter users. Once a user discovers an active pirate repository via a search engine, they only need to interact with the raw IP address once. By utilizing the browser's native bookmarking feature or pinning the tab.

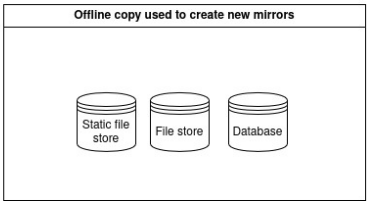
From that point forward, the user experience mimics that of any mainstream website. The user clicks a saved link, the browser connects directly to the pirate server's IP address, and the absence of a formal domain name becomes entirely invisible to the end user. This creates a highly resilient loop: the pirate avoids the paper trail and vulnerability of registering a domain, while the consumer maintains uninterrupted, direct access to the illicit material.



We have searched the query *filmai to* and Google has returned this website with the IP address `45.134.173.224` now users can bookmark this website and on subsequent visits they'll be able to directly navigate to it through the bookmark.

The process of cloning a website





This simple diagram shows one possible configuration of the layout possibly used by the developers.

Building the website with existing APIs

We can get inspired by the cineby.sc website as they explicitly provide us with a very easy way of setting up our own website.

As the first step we obtain our domain names, and point them to our webserver. The webserver might be either buying a dedicated server or alternatively we might use one of the serverless services as demonstrated in the previous section.

After we build our frontend part we have to obtain a TMDB[8] API key[9] to use as a database for searching movies, TV shows, obtaining posters and other metadata. TMDB also gives us IDs, unique identifiers, we can use to obtain the actual pirated content from vidking.net, a snapshot can be found here.

This concludes the development of our websites. Users can now search and watch pirated content. When our domain is taken down, servers seized or IP banned we can just deploy this one frontend server, point one of our remaining domains to it and announce the change to our community.

With modern tooling and Large Language Models (LLM) we might be able to go from zero to fully deployed website in a matter of hours as it allows us to mostly skip the development part of our website.

Building the website from scratch

This setup is slightly more involved but the first few steps are the same, we have to obtain a domain name, this time we would use more than one server, at least 3 or 4. The first one would be our proxy, this server routes all of our traffic and lets us hide the rest of our setup behind it.

For the frontend portion we could use the same setup as in the previous approach.

The difference would be how we handle the content. Our backend would still probably use TMDB but we would store all of the response ourselves as to create a local copy we can then later mirror, this also removes a weak point of TMDB banning us. For this we would need a database and some form of storage. We've talked about S3. It is a perfect fit because we can use the same solution for the actual pirated content.

The last step is to obtain and upload our pirated content, as this is not the point of this study we are not going to delve deeper into this topic.

Now when again our servers are seized, domains taken down or IP addresses banned we can just upload our local backups. The main bottleneck with this approach is that the

[8] TMDB is a popular database of movies and TV shows.
[9] An Application Programming Interface (API) key is a secret unique identifier used to authorize and authenticate users, developers or a program calling to an API.

restoration of our service depends on the size of our library. Typically the frontend and backend services can be up within minutes, the database usually is going to be fairly small and restoring is a very quick process. The biggest hurdle is going to be restoring the pirated content as it can be hundreds of Gigabytes or even multiple Terabytes big. However we can build in a mechanism which would allow us to start the website with no content and prioritize uploading the most popular content first. This way most users would experience very little downtime and all they would have to do is to access the website under a different domain name.

The hybrid approach

We have analyzed many already existing websites and most of these show signs of a hybrid approach. This means they have their own frontend, backend and a database which is also used for storing information such as user accounts, last watched, etc and a third party provider for the content. When a provider is taken down the users have access to many more different providers experiencing virtually zero down time. When the user facing part of the website is taken down all they have to do is to switch to any other mirror as we've noted before.

To summarize all of the research, we can safely claim that once we have the website programmed, spinning up a new mirror is extremely easy and fast.

Step by Step of the cloning process

Once we understand all the possible services these websites use we can describe the process and clearly demonstrate how simple it is to deploy it.

Programming the website

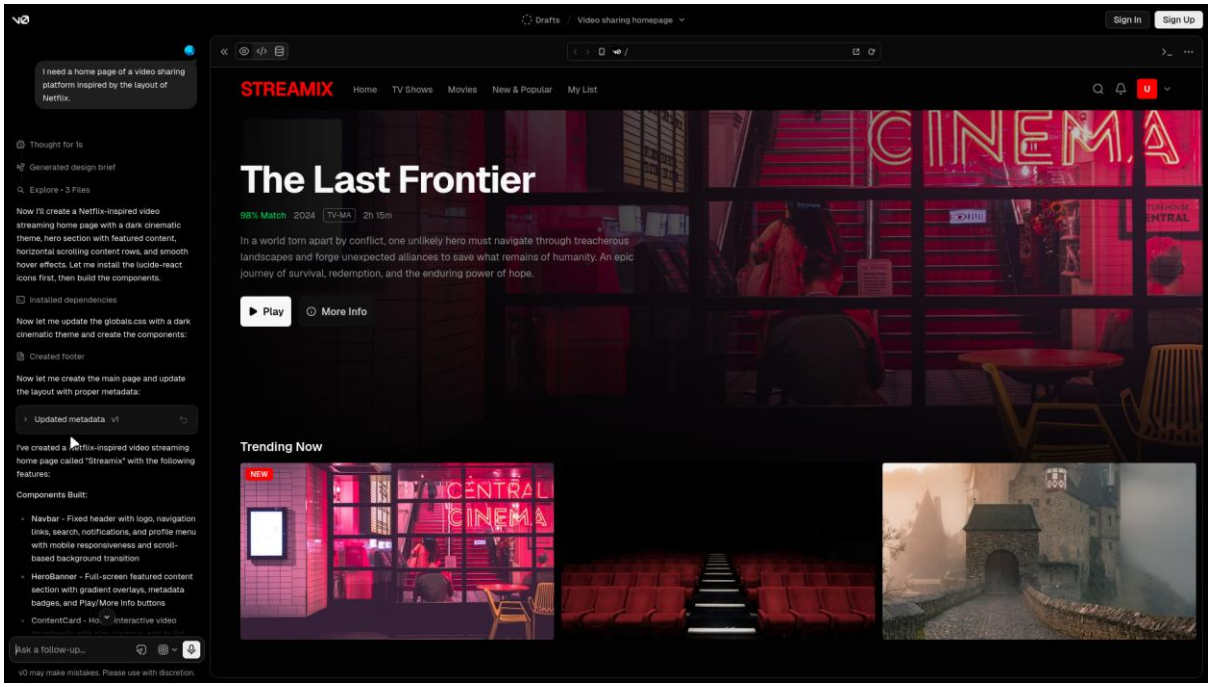
Time complexity: Hours to days, depending on the needed features

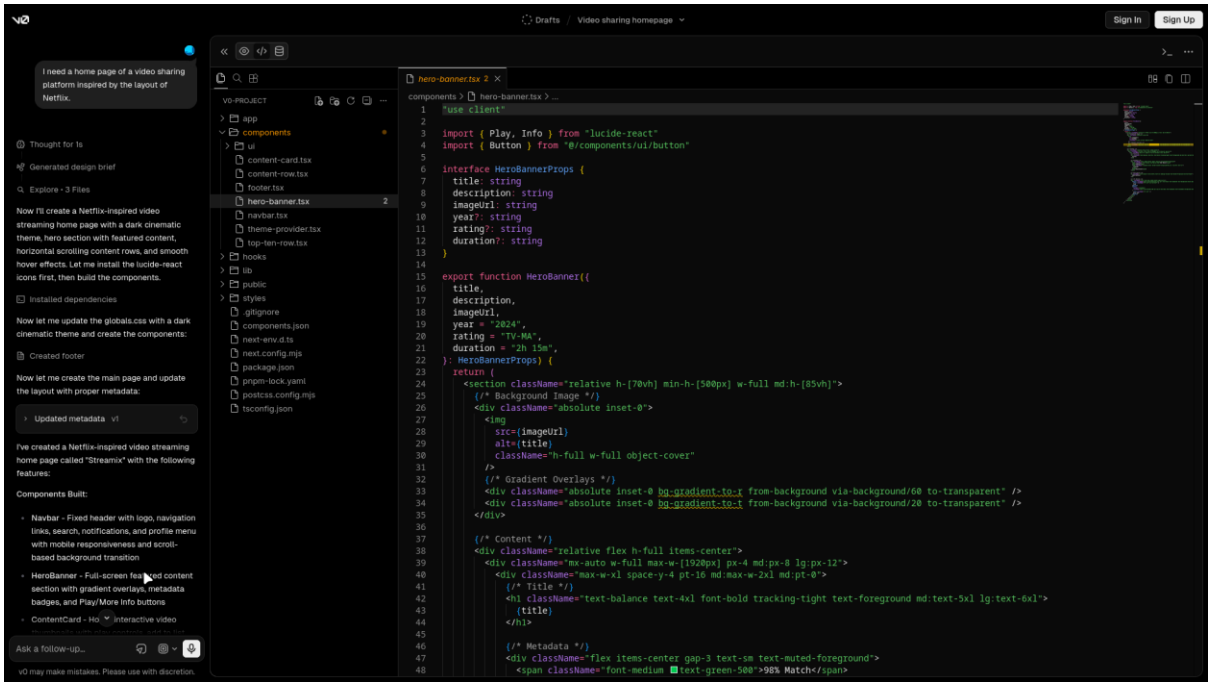
It is not the point of the study to teach which programming languages to use, or how programming works at all. Programming a website manually can be a somewhat complex task but with modern tooling, libraries of components, examples, tutorials and extensive documentation it can be done in a matter of days from completely no code to fully functioning website.

But it is also possible to create most, if not all, of the code with tools like LLMs.

Generating a website

Time complexity: Minutes to hours, depending on the needed features





Generating a fully frontend code is extremely simple with modern tools. The above examples are from [v0.app](v0.app), a free platform for generating websites created by Vercel.

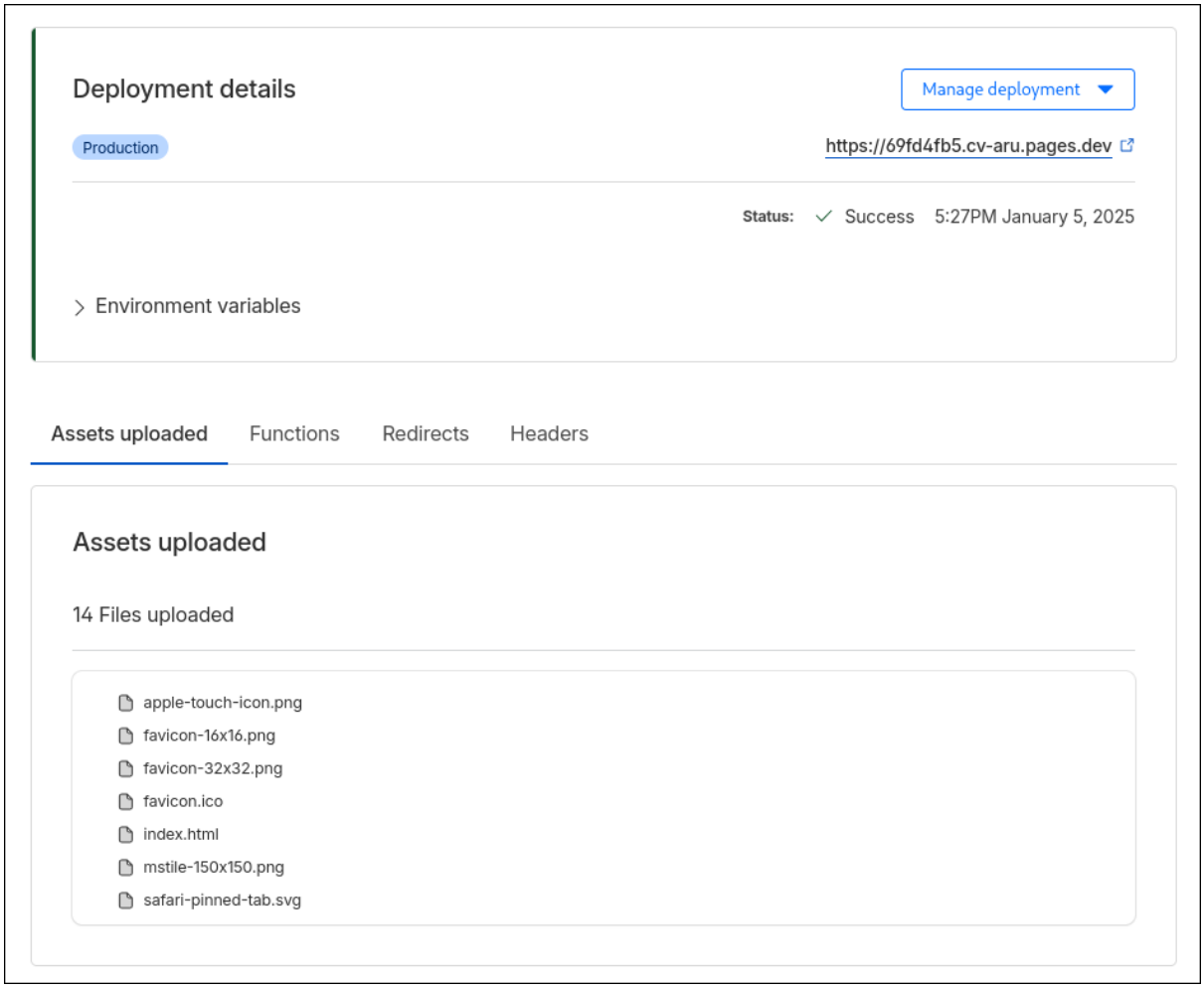
The whole process is as simple as giving it a prompt and waiting for a few minutes. In our example we used *I need a home page of a video sharing platform inspired by the layout of Netflix.* as our prompt and it took around 3 minutes to create this template. Once the initial layout is generated users can ask follow up questions, such as implementing features for

obtaining of the content, user accounts, etc. All of this is now possible without ever interacting with the code and all in plain English.

Deploying the website

Time complexity: Seconds to minutes

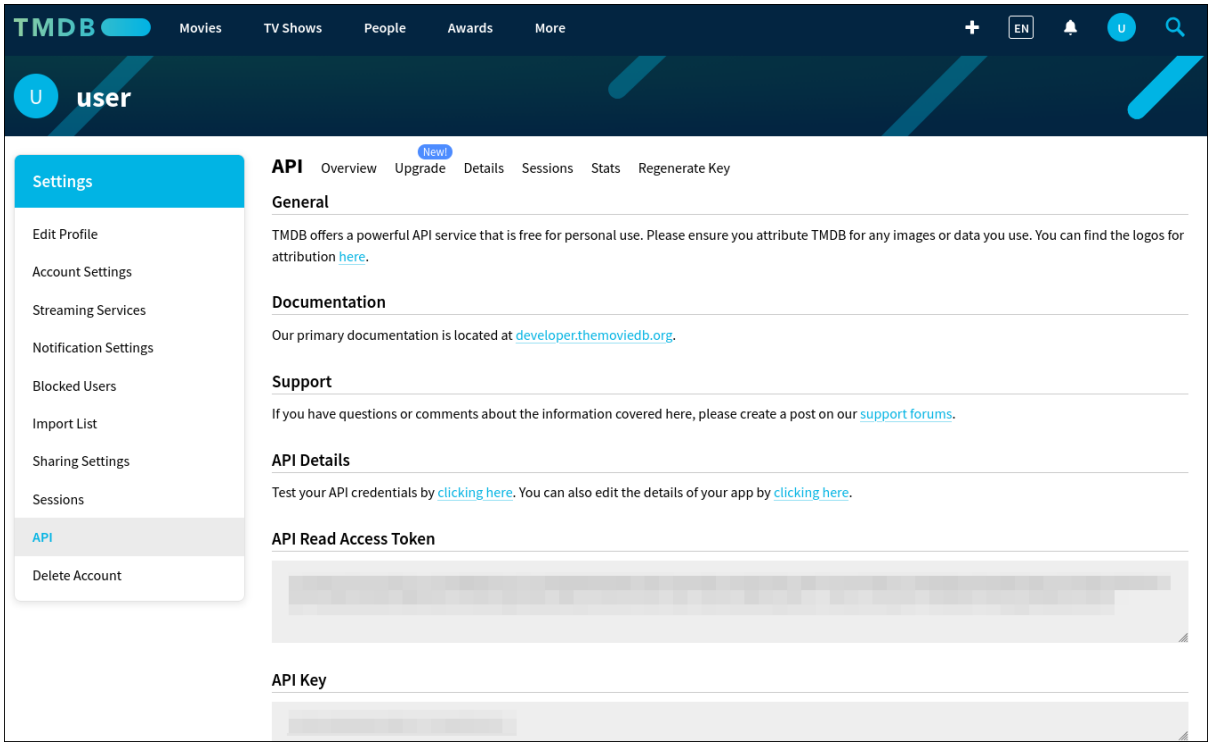
We have already talked about the Pages service of Cloudflare so we are going to use it again, but platforms like Vercel, Netlify and others offer basically the same workflow with just slightly tweaked coats of paint. The developer creates a new website in the User Interface (UI), uploads the source code and within a couple of minutes the website is ready to be accessed.



Third party services

Time complexity: Minutes, at most an hour depending for example on the verification process of the services

If the developer is using the TMDB service (or any other similar service) they need to generate new tokens through the UI.

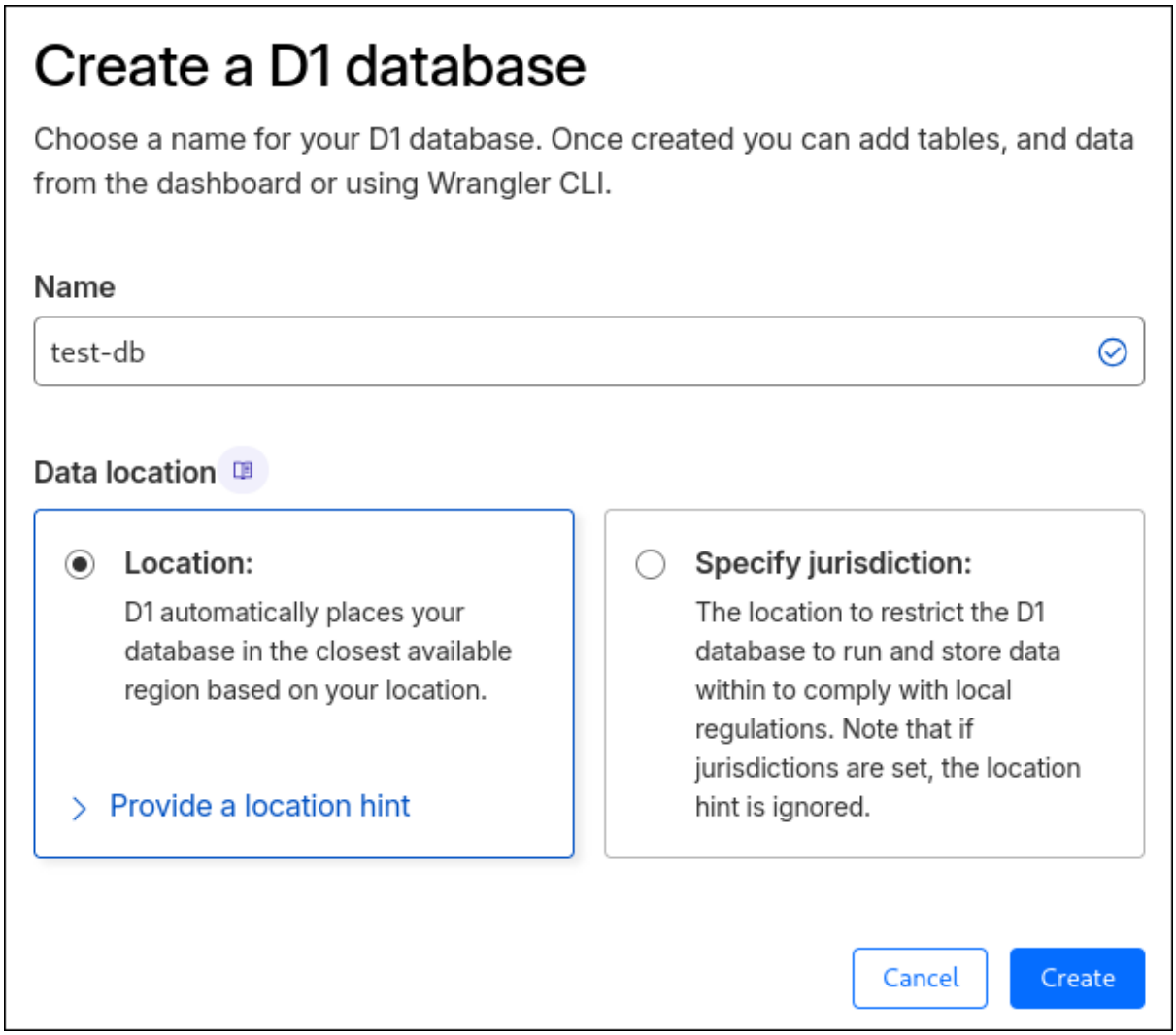


The developers might for example use other services, like generating custom avatars for their users, the flow is typically very similar.

Deploying a database

Time complexity: Minutes, at most an hour if synchronizing large quantities of data

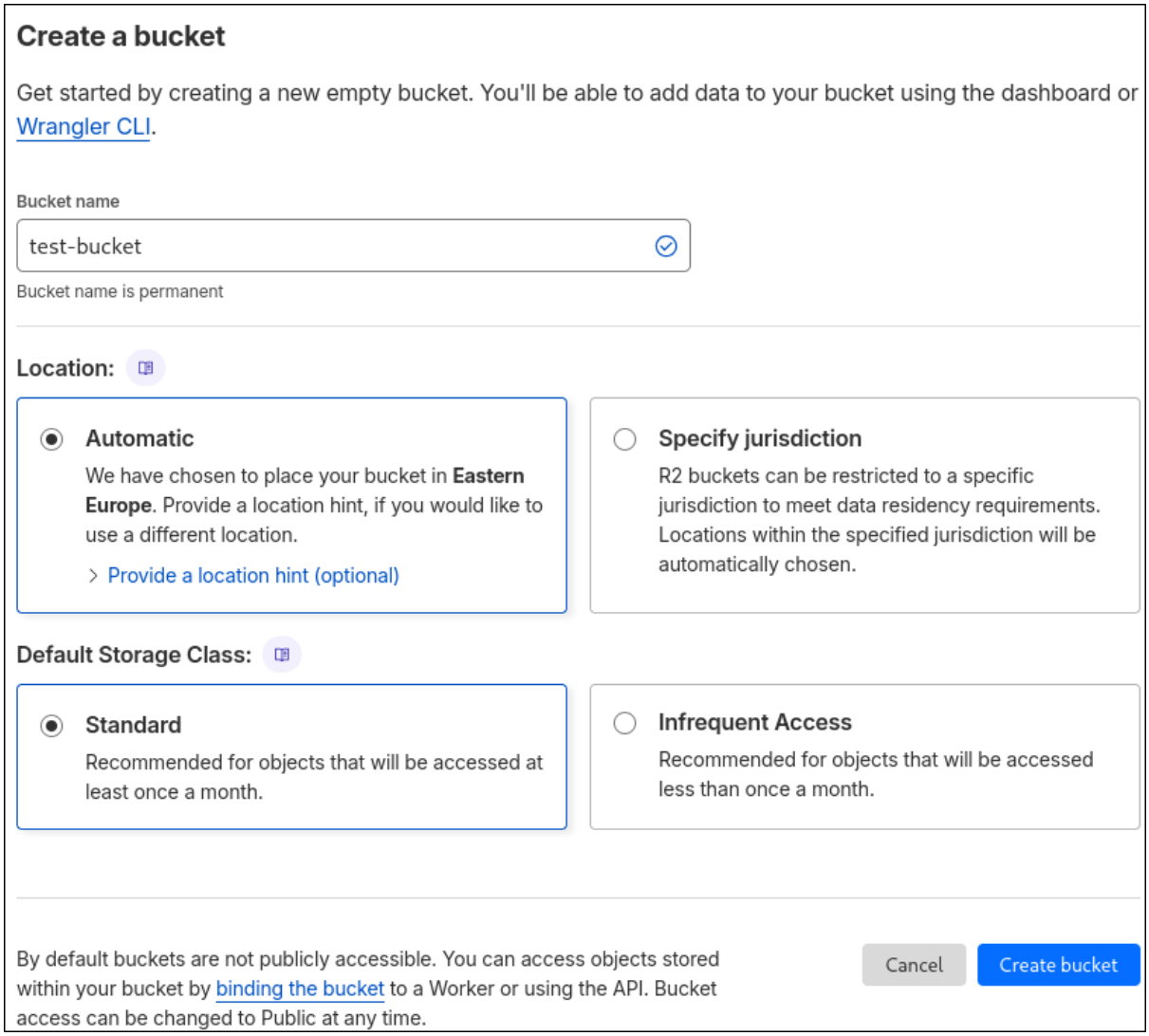
Creating a database is as easy as inputting the name into the UI. This step is very fast and easy, synchronizing the data from a local copy might take some time, typically databases aren't very big so even with slow internet connection it is not going to take significant amount of time.



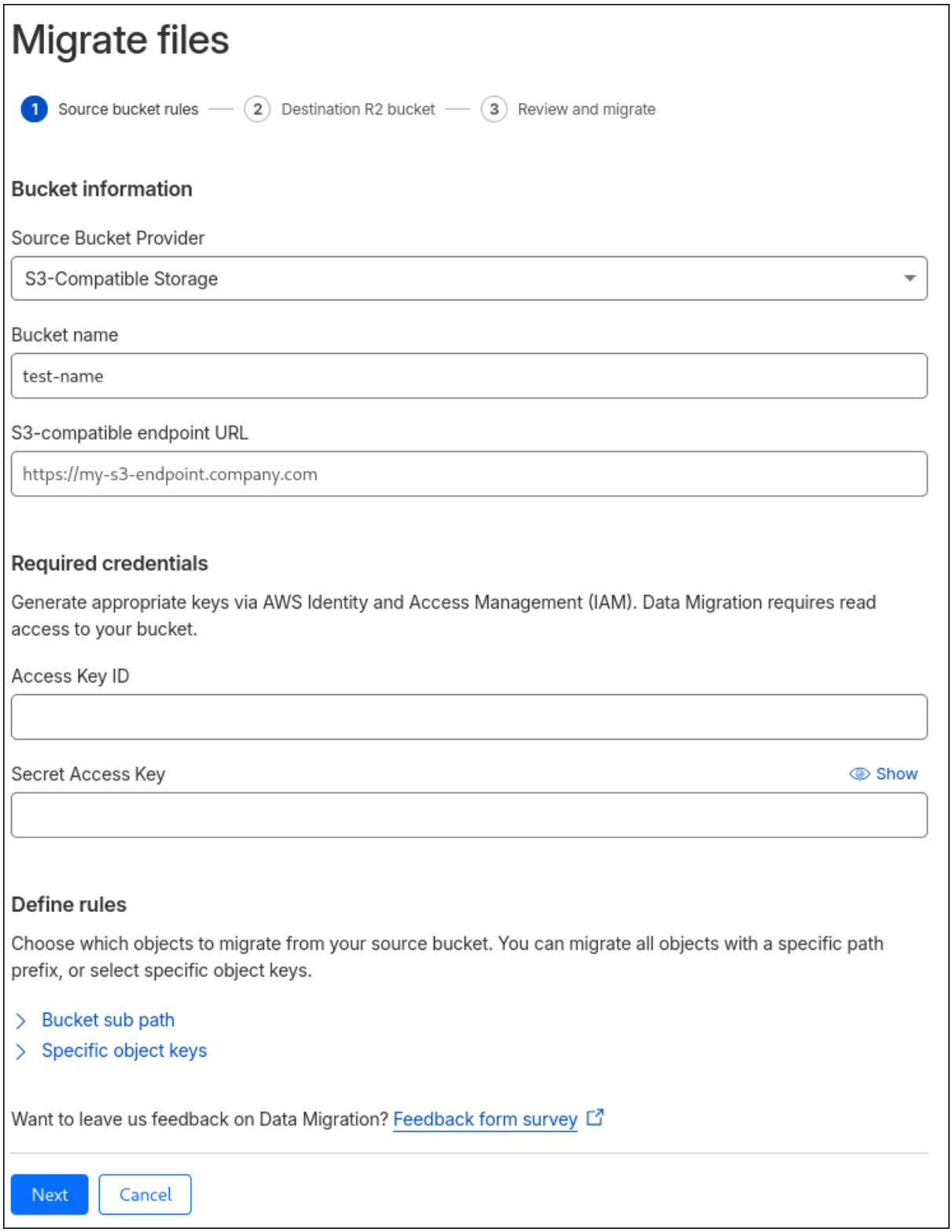
Deploying storage

Time complexity: Hours to days, depending on the quantity of data

For the content the developer would use Cloudflare's R2 service and they have two options. When they start the website for the first time they wouldn't have any of the content so they'd just create an empty bucket. This is again extremely easy as all it requires is to input the name into the UI.



But once the developer has built up their library of content they would typically keep a local copy. This local copy would probably be hosted on some hidden server inaccessible to the internet so as not to attract attention from anyone. Then they would use the migration tool provided by Cloudflare. This tool just asks for the credentials of the user's copy and handles everything automatically.



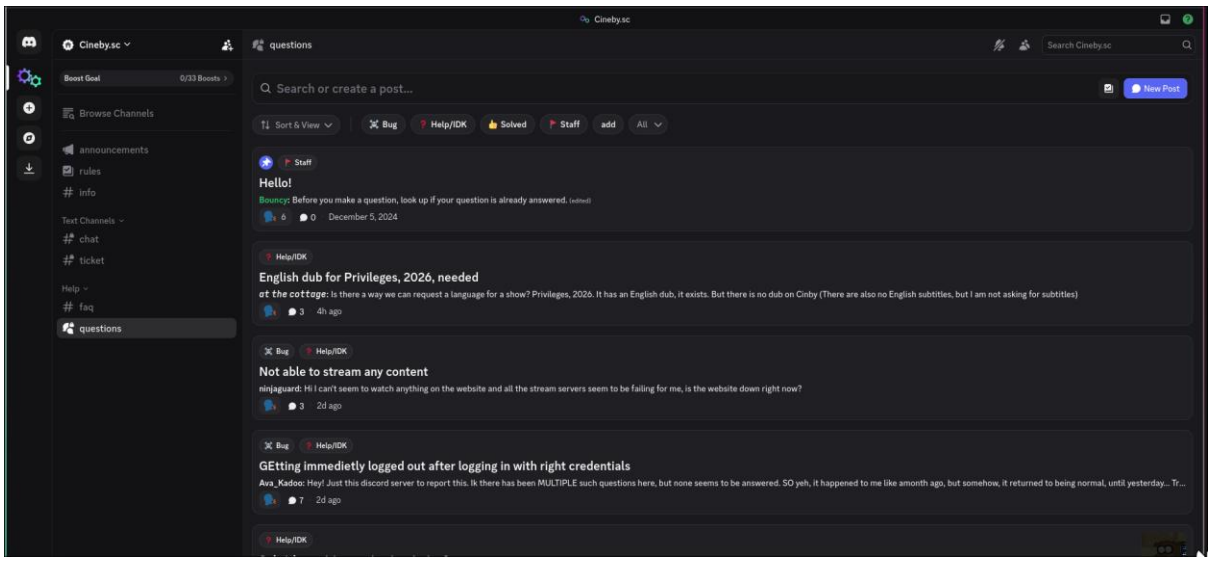
After all these simple steps the developer has created a perfect copy of the website with just a changed domain name. As clearly demonstrated deploying a new copy is done mostly through the UI, no programming skills required. Cloudflare and all of the similar services offer extensive documentation, how-tos and tutorials. Even a non-technical person, if provided with the source code, should be able to deploy their own website.

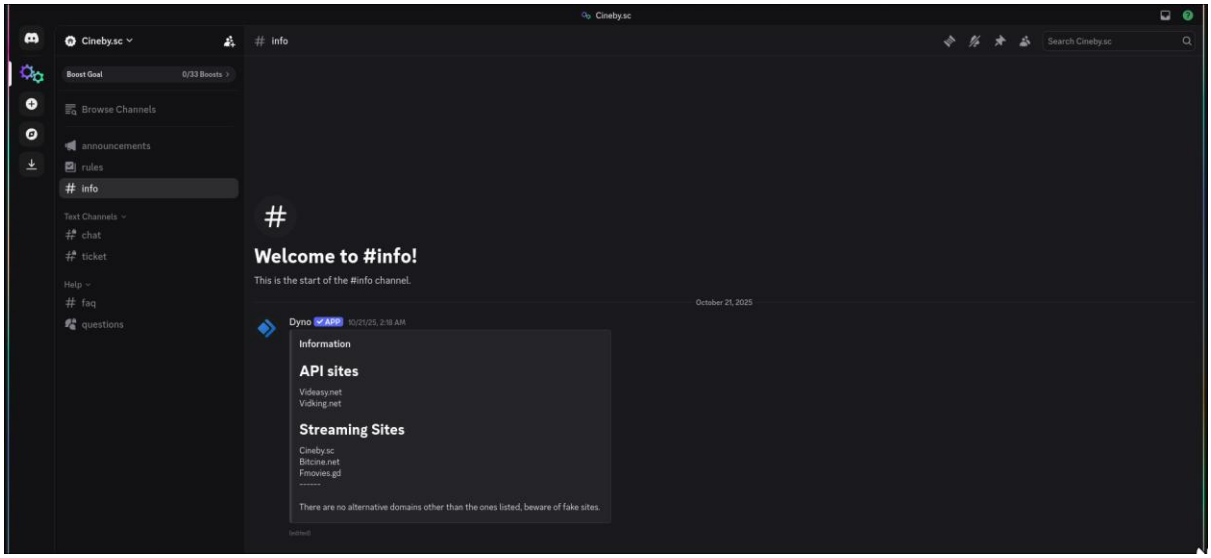
Community

When a website is taken down by internet service providers (ISPs) or copyright enforcement, timing is critical to retaining the user base. Telegram channels act as a direct mass-notification system.

Operators can blast an alert to hundreds of thousands of subscribers simultaneously, announcing the new domain within minutes of a crash. This minimizes user loss and bypasses the slow process of waiting for new domains to rank on search engines.

By shifting the entry point from public search engines to instant messagings platforms such as Discord or Telegram, online piracy networks have effectively decoupled their user base from their web infrastructure. The website domain becomes temporary and disposable, while the Telegram community remains the permanent, un-indexed anchor.





These communities usually have help sections, frequently asked sections and information/announcement sections for notifying users about new domain names, features or any other changes to their websites.

Automatic mirroring using AI

When the website stops working due to a domain name seizure, IP blocking or any other reason the website's owners don't really have to be active to deploy new mirrors.

Monitoring

To know when a website is down a tool called an uptime monitor is used, this tool continuously monitors the health of the services the website is using and can trigger notifications or certain events when a health status changes.

There are many viable options, it is possible to setup an external script or use tools like UptimeRobot, BetterStack or deploy some other alternative on another server. If it detects a non-200 OK status, a timeout, or a DNS resolution failure, it triggers an alert or it can directly trigger the creation of a new mirror.

Deployment

When a mirror gets taken down and the notification is triggered we can configure to have our website automatically deployed. Typically developers use Version Control Systems (VSC) to help manage the development of their websites. GitHub is a very well known service which not only allows the developers to store and manage their code privately but they can also use it to listen for the notifications and run an automated script which is going to deploy a new mirror.

We have used Cloudflare for our examples and they offer a Command Line Interface (CLI) tool which effectively lets developers define the infrastructure with code.

```
curl "https://api.openai.com/v1/responses" -d '{ "input": "..."}'
curl  --url "https://api.cloudflare.com/.../registrar/registrations"
wrangler r2 bucket create my-app-storage-mirror
wrangler d1 create my-app-db-mirror
wrangler d1 execute my-app-db-mirror --file=./schema.sql
wrangler pages deploy ./dist --project-name="my-app-mirror"
wrangler pages domain set my-app-mirror "generated_domain"
curl "https://discord.com/api/v10/interactions/..."
```

The above script in 8 steps, generates a new domain name, registers it using Cloudflare API, starts a new storage bucket, the next two lines create a new database and apply the schema (a description of the layout of the database, which tables to use, names of columns, etc). The next two lines deploy the actual website and link a domain to it. The

last line in our example broadcasts the new mirror to the discord channel so users can switch to it.

Deploying like this is mostly fully automated and very quick. For this to work the developers would typically have multiple Cloudflare accounts ready and the script would just pick which account to use.

To go more into details the script has basically three phases.

1. **Obtain domain:** As you can see the first two lines use the `curl` command to contact OpenAI API to generate a new domain name and then contact Cloudflare and use the AI output to register a new domain name. (In reality it would be a good idea to generate more names, check their availability before registering).
2. **Setup infrastructure:** The rest of the script then proceeds to setup the storage, database, upload the database schema, upload the website code and link the newly obtained domain.
3. **Broadcast change:** As the last step the Discord API is used to broadcast the new domain name of the new mirror to users.

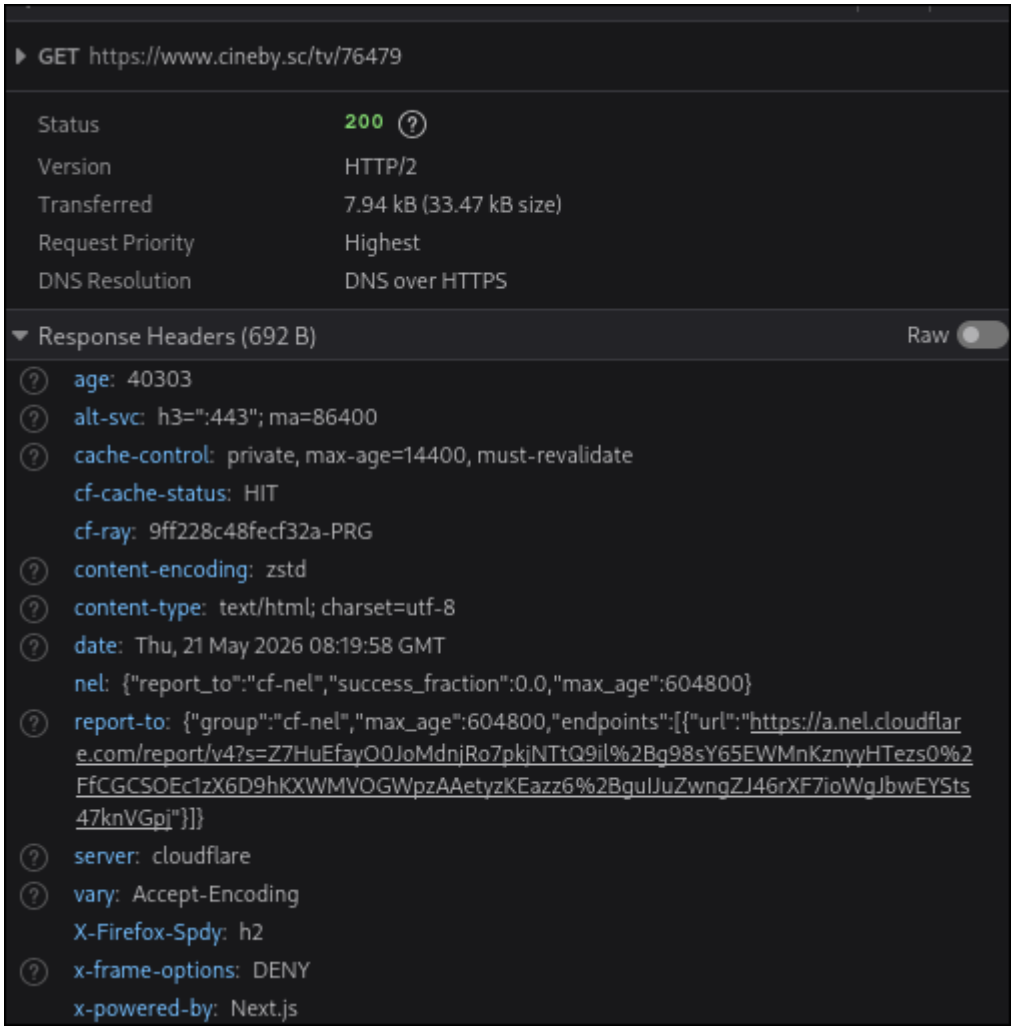
Clearly all of these steps allow, with also using an uptime monitor, to deploy new mirrors with new unique domain names without the developers ever having to do any manual work.

Case Study: Cineby

Cineby is a great example of all of the techniques we've described above. It has multiple mirrors and uses multiple services.

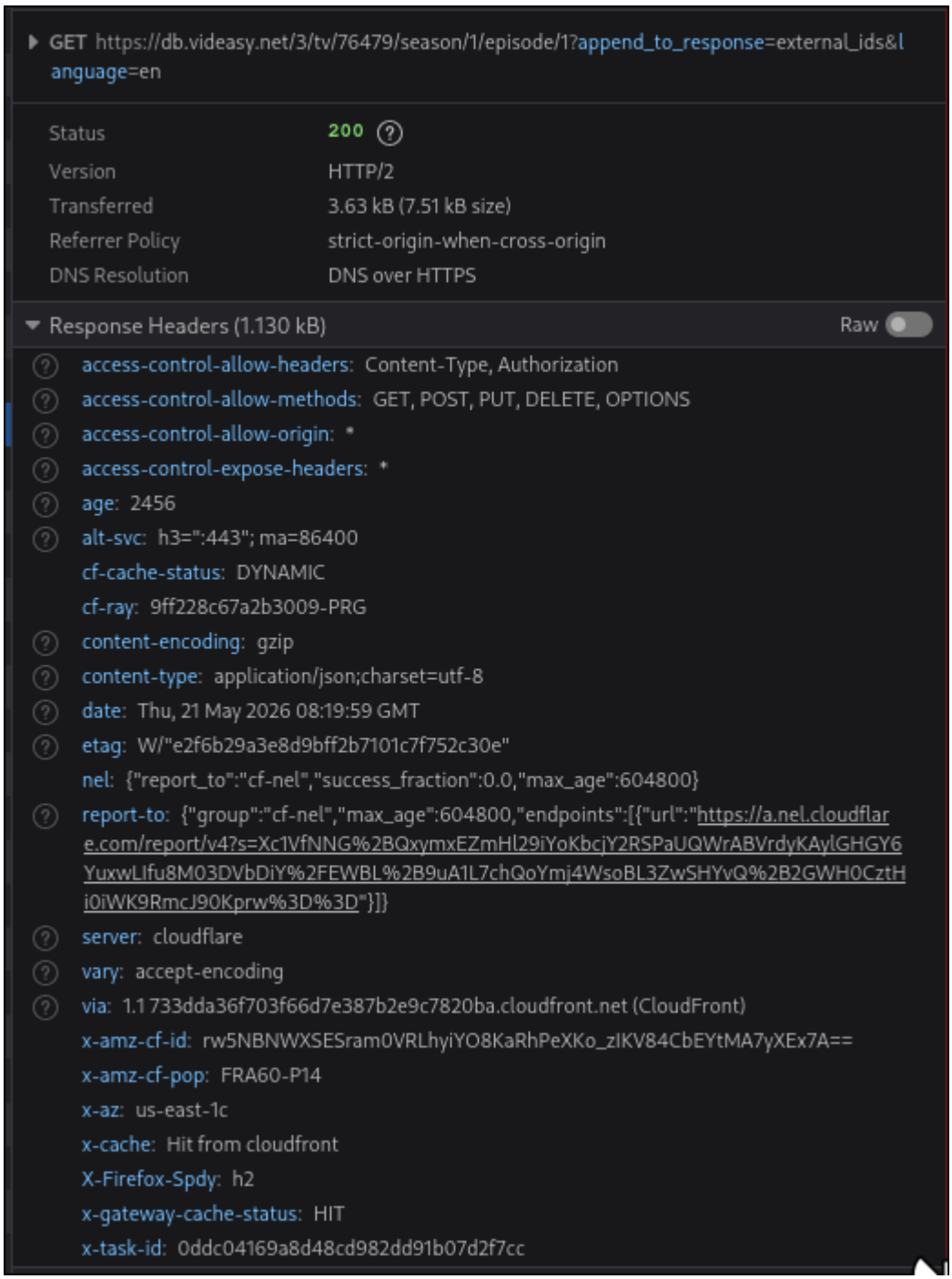
The main mirror is cineby.sc and bitcine.tv is the second one. The current state of cineby.sc and bitcine.tv is backed up using Wayback Machine.

We will be looking at cineby.sc and describing how it works under the hood. Users interact with a frontend server which serves them the visuals, the website is written using Next.js a framework created by Vercel. We have mentioned this company before when we demonstrated how easy it is to deploy a website and it is entirely possible that this website is using Vercel for hosting. From digging through network requests we get both confirmation the site is powered by Next.js but also that they are using Cloudflare to mitigate attacks from automated bot networks. Note the x-powered-by and report-to lines.



The next service used comes from the domain db.videasy.net and is used to obtain information about the content, such as the name, crew working on the original work or the air date, current snapshot can be found here.

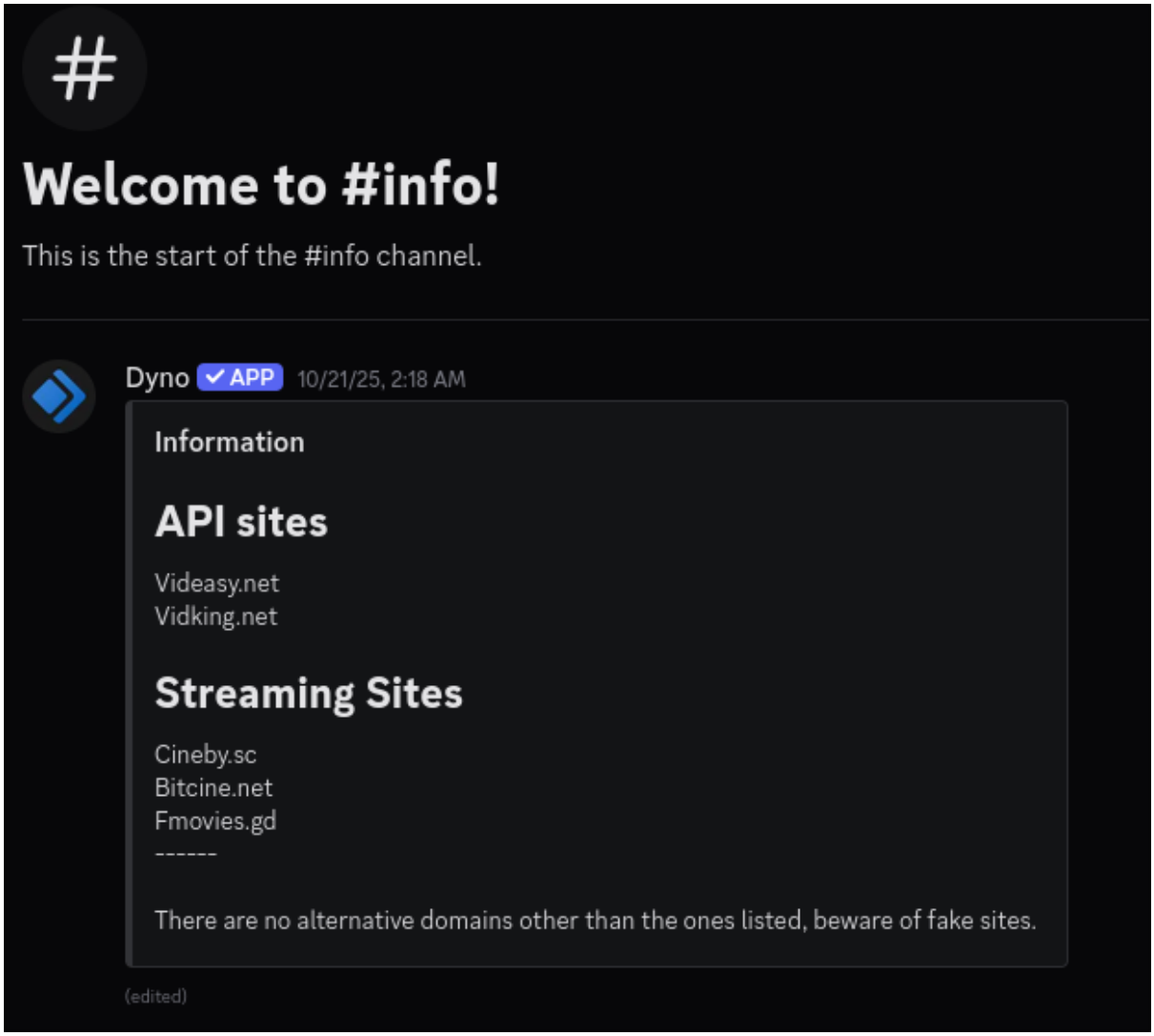
Again using the network request we can see that this time this service is hosted using the Amazon Web Service (AWS) platform, note the x-cache or x-az lines.



As the last step the website has to source the video content from somewhere and for this the easy.speedsterwave.app domain is used. We can see using the network request the browser loading the necessary video files.

The previously mentioned vidking.net is used as a player, a snapshot can be found here, and this service is sourcing the content from easy.speedsterwave.app again proving modern piracy websites use multilayered infrastructure.

As the final piece of the puzzle is the community hosted on discord.com and we can see that the *info* channel is used to share information about available mirrors with the community.



Overview of cineby.sc

Users wanting to use this site join the discord server, check the *info* channel for available websites.
When interacting with the website they communicate with a frontend server. The frontend server is using another service to source information about available content such as images, crew members, names of episodes, etc.
Alongside this information a unique identifier is returned which in turn is used to embed a player from yet again different service. This service is under the hood sourcing video data from some storage service which tries to lay low as possible and it does not even have their own accessible website.

This setup helps illustrate that taking down these services is very hard as they are mostly independent and to truly stop them from existing it would require to takedown the discord

server, the front end server, the service for hosting information about content, the embedded player and the storage service at the same time.

Conclusion

This study demonstrates that modern digital piracy has evolved from amateur single server operations to highly resilient, multi service effectively tech startups. Driven by automation, containerization and advanced tooling based on AI, launching and maintaining these streaming platforms has become straightforward, even for individuals with minimal technical expertise.

Even though DNS/IP blocking is still an effective technique, it creates immediate friction, strips away public-facing domain names, and temporarily disrupts traffic, the findings of this research make it undeniable that these perimeter blocks are no longer enough to completely halt piracy networks.

Because modern piracy networks employ highly decoupled, cloud-native architectures, a block at the DNS or IP level is treated as a minor inconvenience rather than a fatal blow. As soon as an ISP implements a block, automated infrastructure can instantly spin up a new web proxy or mirror site under a completely different domain. Crucially, by migrating their user bases to alternative, highly resilient communication channels like Discord or Telegram, piracy operators have permanently unlinked their audiences from their temporary web addresses. When a domain vanishes, automated bots immediately route hundreds of thousands of users to an active mirror within minutes, ensuring near-continuous uptime. Furthermore, tactics like direct IP indexing allow search engine crawlers to map and deliver active sites straight to consumers using raw numerical strings, completely bypassing the DNS block structure altogether.

Ultimately, relying on legacy blocking methods is akin to treating the symptoms of a rapidly mutating digital epidemic rather than curing it. These reactive enforcement mechanisms fail to address the core issue: the severe technological asymmetry between piracy networks and enforcement bodies.