

P4 - CDN & Docker

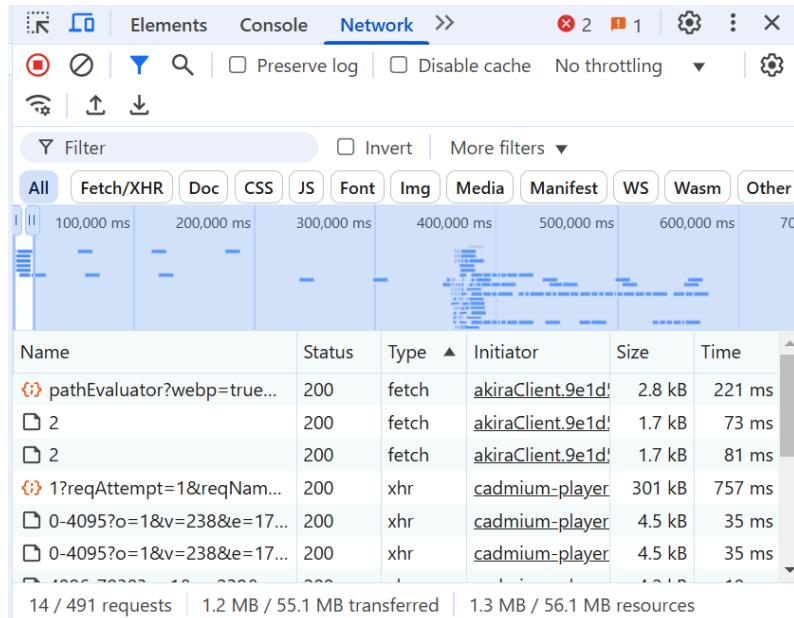
1. Download Docker for the command line or for Desktop and install it.

```
PS C:\Users\isall\OneDrive\UNI\4_uni\P4_Vid3o> ffmpeg -version
ffmpeg version 2024-09-26-git-f43916e217-full_build-www.gyan.dev Copyright (c) 2000-2024 the FFmpeg developers
built with gcc 13.2.0 (Rev5, Built by MSYS2 project)
configuration: --enable-gpl --enable-version3 --enable-static --disable-w32threads --disable-autodetect --enable-fontconfig --enable-iconv --enable-gnutls --enable-libxml2 --enable-gmp --enable-bzlib --enable-lzma --enable-lbsnappy --enable-zlib --enable-librist --enable-libssh --enable-libzmq --enable-avisynth --enable-libbluray --enable-libcaca --enable-sdl2 --enable-libaribbb24 --enable-libaribcaption --enable-libdavid --enable-libdavfs2 --enable-libopenjpeg --enable-libquirc --enable-libuavs3d --enable-libxevd --enable-libzvbi --enable-libqrencode --enable-librav1e --enable-libsvtav1 --enable-libvvenc --enable-libwebp --enable-libx264 --enable-libx265 --enable-libxavs2 --enable-libxevc --enable-libxvid --enable-libaom --enable-libjxl --enable-libvpx --enable-medialibrary --enable-libass --enable-frei0r --enable-libfreetype --enable-libfribidi --enable-libharfbuzz --enable-liblensfun --enable-libvidstab --enable-libvmaf --enable-libzimg --enable-amf --enable-cuda-llvm --enable-cuvid --enable-dxva2 --enable-d3d11va --enable-d3d12va --enable-fnvcdec --enable-libvpl --enable-nvdec --enable-nvenc --enable-vaapi --enable-libshaderc --enable-vulkan --enable-libplacebo --enable-opencl --enable-libcdio --enable-libgme --enable-libmodplug --enable-libopenmpt --enable-libopencore-amrwb --enable-libmp3lame --enable-libshine --enable-libtheora --enable-libtwolame --enable-libvo-amrwbenc --enable-libcodec2 --enable-libilbc --enable-libgsm --enable-liblrc3 --enable-libopencore-amrnb --enable-libopus --enable-libspeex --enable-libvorbis --enable-ladspa --enable-libbs2b --enable-libflite --enable-libmysofa --enable-librubberband --enable-libsoxr --enable-chromaprint
libavutil      59. 40.100 / 59. 40.100
libavcodec     61. 20.100 / 61. 20.100
libavformat    61.  8.100 / 61.  8.100
libavdevice    61.  4.100 / 61.  4.100
libavfilter     10.  5.100 / 10.  5.100
libswscale       8.  4.100 /  8.  4.100
libswresample    5.  4.100 /  5.  4.100
libpostproc     58.  4.100 / 58.  4.100
PS C:\Users\isall\OneDrive\UNI\4_uni\P4_Vid3o>
```

2. d
3. d

4. We have connected into *Netflix*, a VOD platform. Then, we used the *F12* command to open *Developer Tools*, navigated to the *Network* tab, and examined the *Type* column.

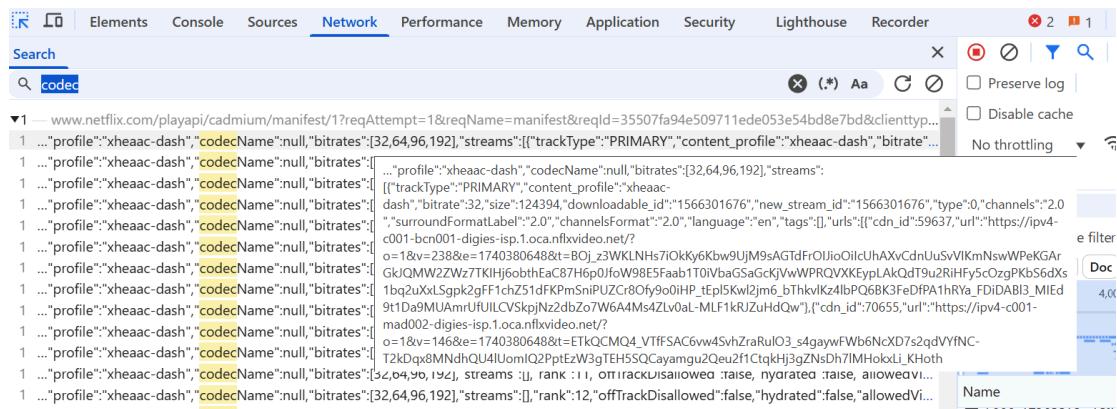
Then we have click play “*The Fast and the Furious*” and observe the following:



We see that the video is not downloaded as a single file but in small chunks. This indicates that Netflix uses **DASH**, a system that adjusts video quality based on internet speed to prevent buffering or interruptions during playback.

By accessing one of the types, we found the following url: <https://www.netflix.com/ng/website/memberapi/release/pathEvaluator?webp=true&drmSystem=widevine...> which includes the term "widevine." Widevine is a DRM system developed by Google, confirming that Netflix uses DRM to protect its content.

In order to find the type of codec, we have search for it in the search bar, and have found:



But we can see that it says that it is “*Null*”. So Netflix is not giving us the details. We have searched for other ways to find it, but still no answer. So we searched on the internet and found out that Netflix uses Av1.

5.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

```
mariasanninomezquita -- zsh -- 80x24
To start nginx now and restart at login:
brew services start nginx
Or, if you don't want/need a background service you can just run:
/opt/homebrew/opt/nginx/bin/nginx -g daemon off!;
=> Summary:
  0 /opt/homebrew/Cellar/nginx/1.27.4: 27 files, 2.5MB
=> Running 'brew cleanup nginx'...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see 'man brew').
mariasanninomezquita@Air-de-Maria-2 ~ % brew services start nginx
=> Tapping homebrew/services
Cloning into '/opt/homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 4085, done.
remote: Counting objects: 100% (520/520), done.
remote: Compressing objects: 100% (186/186), done.
remote: Total 4085 (delta 413), reused 338 (delta 334), pack-reused 3565 (from 2 )
Receiving objects: 100% (4085/4085), 1.23 MiB | 1.34 MiB/s, done.
Resolving deltas: 100% (3984/3984), done.
Tapped 2 commands (53 files, 1.4MB).
=> Successfully started 'nginx' (label: homebrew.mxcl.nginx)
mariasanninomezquita@Air-de-Maria-2 ~ %
```

README BSD-3-Clause license
you should see the json response :)

Adding caching capabilities

For the backend service to be cacheable we need to set up the caching policy. We'll use the HTTP header [Cache-Control](#) to setup what caching behavior we want.

```
-- we want the content to be cached by 10 seconds OR the provided max_age (ex: nginx.header['Cache-Control'] = 'public, max-age=' .. (nginx.var.arg_max_age or 1f)
```

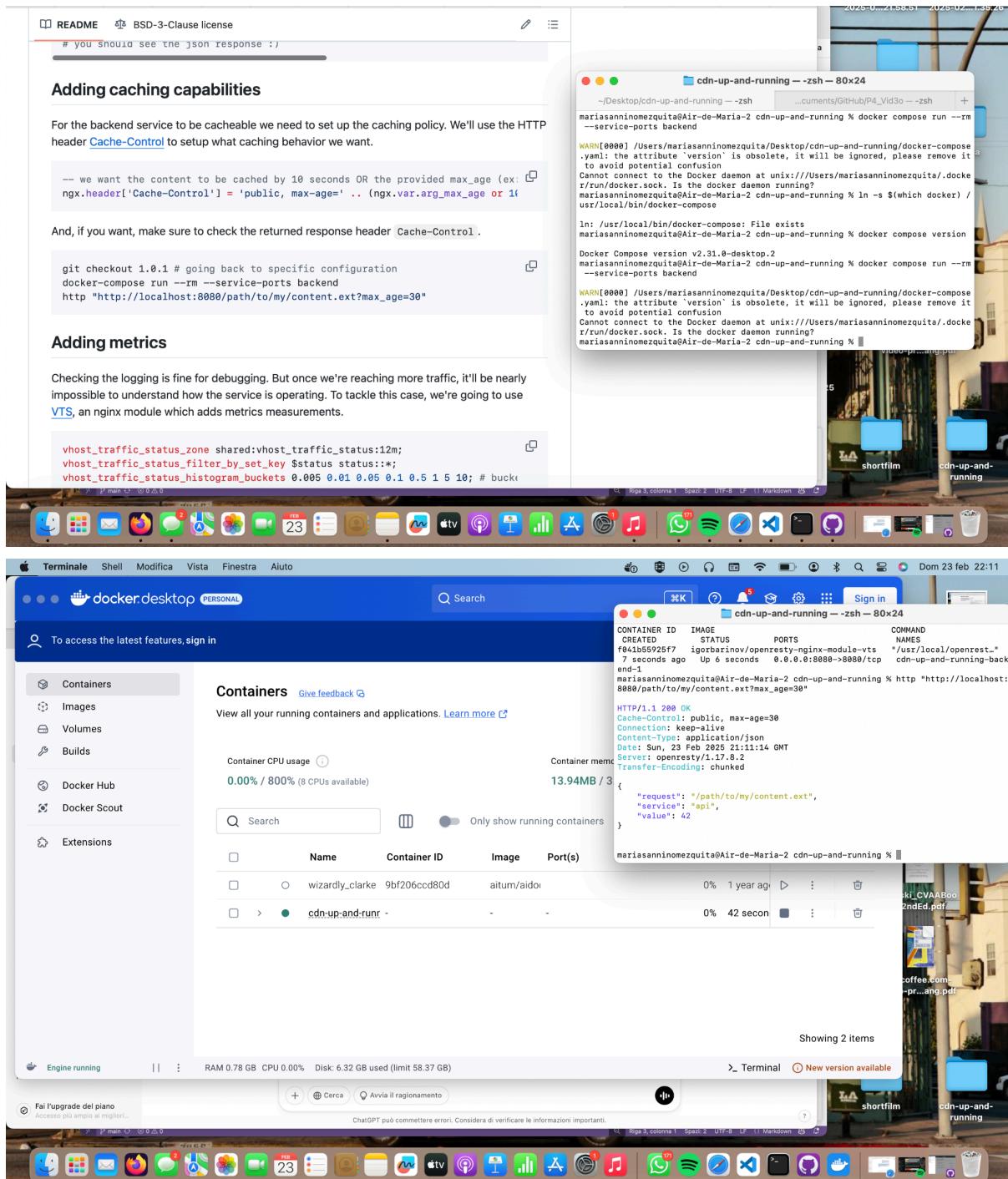
And, if you want, make sure to check the returned response header [Cache-Control](#).

```
git checkout 1.0.1 # going back to specific configuration
docker-compose run --rm --service-ports backend
http "http://localhost:8080/path/to/my/content.ext?max_age=30"
```

Adding metrics

Checking the logging is fine for debugging. But once we're reaching more traffic, it'll be nearly impossible to understand how the service is operating. To tackle this case, we're going to use [VTS](#), an nginx module which adds metrics measurements.

```
vhost_traffic_status_zone shared:vhost_traffic_status:12m;
vhost_traffic_status_filter_by_set_key $status status:*=;
vhost_traffic_status_histogram_buckets 0.005 0.01 0.05 0.1 0.5 1 5 10; # buckets
```



The screenshot shows a Mac desktop environment with several open windows:

- Terminal Window:** Displays command-line output related to Docker and Nginx configuration, including logs from Docker Compose and Nginx.
- Docker Desktop:** Shows the Docker Hub interface with options like "Containers", "Images", "Volumes", "Builds", "Docker Hub", "Docker Scout", and "Extensions". A container named "cdn-up-and-running" is selected, showing its status (Up 6 seconds), ports (8.0.0.0:8080->8080/tcp), and command line.
- Browser Window:** Shows a simple Nginx configuration page with sections for "Adding caching capabilities" and "Adding metrics". It includes code snippets for Docker Compose and Nginx configuration files.

```

README  BSD-3-Clause license

VTS, an nginx module which adds metrics measurements.

vhost_traffic_status_zone shared:vhost_traffic_status:12m;
vhost_traffic_status_filter_by_set_key $status status:::;
vhost_traffic_status_histogram_buckets 0.005 0.01 0.05 0.1 0.5 1 5 10; # buckets

The vhost_traffic_status_zone sets a memory space required for the metrics. The vhost_traffic_status_filter_by_set_key groups metrics by a given variable (for instance, we decided to group metrics by $status) and finally, the vhost_traffic_status_histogram_buckets provides a way to bucketize the metrics in seconds. We decided to create buckets varying from 0.005 to 10 seconds, because they will help us to create percentiles (p99, p50, etc).

location /status {
    vhost_traffic_status_display;
    vhost_traffic_status_display_format html;
}

We also must expose the metrics in a location. We will use the /status to do it.

git checkout 1.1.0
docker-compose run --rm --service=ports backend
# if you go to http://localhost:8080/status/format/html you'll see information
# notice that VTS also provides other formats such as status/format/prometheus

{"service": "api", "value": 42, "request": "/"}

zsh: command not found: vhost_traffic_status_zone
zsh: no matches found: status:::*
zsh: command not found: vhost_traffic_status_histogram_buckets
zsh: command not found: #
mariasaninomezquita@Air-de-Maria-2 cdn-up-and-running % vhost_traffic_status_zone shared:vhost_traffic_status:12m;
vhost_traffic_status_filter_by_set_key $status status:::;;
vhost_traffic_status_histogram_buckets 0.005 0.01 0.05 0.1 0.5 1 5 10;
zsh: command not found: vhost_traffic_status_zone
zsh: no matches found: status:::*
zsh: command not found: vhost_traffic_status_histogram_buckets
mariasaninomezquita@Air-de-Maria-2 cdn-up-and-running % cdn-up-and-running/nginx.conf
zsh: no such file or directory: cdn-up-and-running/nginx.conf
mariasaninomezquita@Air-de-Maria-2 cdn-up-and-running % /Users/mariasaninomezquita/Desktop/cdn-up-and-running/nginx_backend.conf
zsh: permission denied: /Users/mariasaninomezquita/Desktop/cdn-up-and-running/nginx_backend.conf
mariasaninomezquita@Air-de-Maria-2 cdn-up-and-running % nano /Users/mariasaninomezquita/Desktop/cdn-up-and-running/nginx.conf
mariasaninomezquita@Air-de-Maria-2 cdn-up-and-running % docker-compose down
docker-compose up -d

```

Nginx Vhost Traffic Status

Server main

| Host | Version | Uptime | Connections | | | | Requests | | | | Shared memory | | | | |
|--------------|---------|--------|-------------|---------|---------|---------|----------|---------|-------|-------|---------------|----------------------|----------|----------|---|
| | | | active | reading | writing | waiting | accepted | handled | Total | Req/s | name | maxSize | usedSize | usedNode | |
| 1284b8bb67e2 | 1.17.8 | 1m 38s | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 9 | 0 | vhost_traffic_status | 12.0 MB | 7.0 KB | 2 |

Server zones

| Zone | Requests | | | | | | | | Responses | | | | Traffic | | | | Cache | | | | | | | | |
|------|----------|-------|------|-----|-----|-----|-----|-----|-----------|-----|-----|-----|---------|-------|------|---------|--------|------|--------|---------|-------|----------|-------------|-----|--------|
| | Total | Req/s | Time | 499 | 500 | 502 | 504 | 1xx | 2xx | 3xx | 4xx | 5xx | Total | Sent | Rcvd | Sent/s | Rcvd/s | Miss | Bypass | Expired | Stale | Updating | Revalidated | Hit | Scarce |
| - | 8 | 0 | 0ms | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 8 | 164.2 | 2.7 | 3.4 KIB | 339 B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| * | 8 | 0 | 0ms | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 8 | 164.2 | 2.7 | 3.4 KIB | 339 B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Filters

status::*

| Zone | Requests | | | | | | | | Responses | | | | Traffic | | | | Cache | | | | | | | | |
|------|----------|-------|------|-----|-----|-----|-----|-----|-----------|-----|-----|-----|---------|-------|------|---------|--------|------|--------|---------|-------|----------|-------------|-----|--------|
| | Total | Req/s | Time | 499 | 500 | 502 | 504 | 1xx | 2xx | 3xx | 4xx | 5xx | Total | Sent | Rcvd | Sent/s | Rcvd/s | Miss | Bypass | Expired | Stale | Updating | Revalidated | Hit | Scarce |
| 200 | 8 | 0 | 0ms | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 8 | 164.2 | 2.7 | 3.4 KIB | 339 B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

update interval: 1 sec

[JSON](#) | [GITHUB](#)

The screenshot shows a Mac desktop environment with several open windows:

- Terminal Window:** Shows a command-line session for a container named "cdn-up-and-running". The session includes commands like "nano /Users/mariasanninomezquita/Desktop/cdn-up-and-running/nginx_backend.conf" and "curl -X GET http://localhost:8080/".
- Docker Desktop:** A central application window titled "Docker Desktop" showing running containers. It lists four containers:

| Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------|---------------|-----------------------|-----------|---------|---------------|-----------|
| wizardly_clarke | 9bf206cccd80d | aitum/aidocker:latest | - | 0% | 1 year ago | [Actions] |
| cdn-up-and-running | - | - | - | 0% | 3 minutes ago | [Actions] |
| nginx_backend | 070efb429740 | nginx:latest | 8080:8080 | 0% | 3 minutes ago | [Actions] |
| nginx_cdn | 71676758cc37 | nginx:latest | 8081:8081 | 0% | 3 minutes ago | [Actions] |
- Browser Window:** A Safari window showing a local file "cdn-up-and-running" which contains a simple HTML page with a single line of text: "Hello from the CDN".
- System Dock:** The Mac OS X dock at the bottom of the screen, containing icons for various applications like Mail, Finder, and Safari.

```

listen 8080;

location / {
    proxy_pass http://backend;
    add_header X-Cache-Status $upstream_cache_status;
}

We also added a new header (X-Cache-Status) to indicate whether the cache was used or not.
• HIT: when the content is in the CDN, the X-Cache-Status should return a hit.
• MISS: when the content isn't in the CDN, the X-Cache-Status should return a miss.

git checkout 2.0.0
docker-compose up
# we still can fetch the content from the backend
http "http://localhost:8080/path/to/my/content.ext"
# but we really want to access the content through the edge (CDN)
http "http://localhost:8081/path/to/my/content.ext"

```

Caching

When we try to fetch content, the X-Cache-Status header is absent. It seems that the edge node is always invariably requesting the backend. This is not the way a CDN should work, right?

```

backend_1 | 172.22.0.4 - - [05/Jan/2022:17:24:48 +0000] "GET /path/to/my/c [redacted]
edge_1   | 172.22.0.1 - - [05/Jan/2022:17:24:48 +0000] "GET /path/to/my/c [redacted]

```

getting an error:

```

leandro.moreira at C02D557BML7L in ~/tmp/cdn-up-and-running (tags/2.0.0)
$ http "http://localhost:8081/path/to/something.txt"
HTTP/1.1 200 OK
Cache-Control: public, max-age=10
Connection: keep-alive
Content-Type: application/json
Date: Thu, 06 Jan 2022 10:17:31 GMT
Server: openresty/1.17.8.2
Transfer-Encoding: chunked
X-Cache-Status: HIT
X-Edge: Server

{
    "request": "/path/to/something.txt",
    "service": "api",
    "value": 42
}

```

Monitoring Tools

Checking the cache effectiveness by looking at the command line isn't efficient. It's better if we use a tool for that. **Prometheus** will be used to scrape metrics on all servers, and **Grafana** will show graphics based on the metrics collected by the prometheus.