Last edited 25 March 2022.

Table of Contents

Last edited 25 March 2022.

Challenges

Customer has been deploying Nginx Open Source on Windows for Video live stream to multiple sites. Existing setup is based on 2 servers pushing the Livestream from data source and this stream will be distributed to 15 sites. It was running smoothly at 40 user end points but latency begins at 2500 user end points. During one of the Live events, users experienced latency to about 1 minute and has screen freezing. Pre-event testing showed good bandwidth and connection as claimed by their video equipment provider.

It seems to be a peculiar case where nobody knows which part of the architecture went wrong.

- Was it the under-powered VMs running NGINX OSS?
- Was it an unofficial windows version of NGINX?
    - That was really a surprised to us! We didn't even knew windows-version existed.
- Was it the hardware needed to use a Solid State Drive?
- Was it the ISP's bandwidth?
- Was it the firewall/network devices becoming bottlenecks?

So many possibilities. But we focused on the current configuration and whether the current architecture can be optimized.

Last edited 25 March 2022.

Fact finding customer's current architecture

Several potential problems were identified after looking at customer's current nginx OSS config files.

<<Diagram not shown due to confidentiality, but you won't miss out a lot as it's a straight-forward traffic flow except the below "issues" we found>>

At the Publisher site (Origin/Source)

- "Stressed-out" publisher has too many servers to 'push' fragments towards,
- Transcoding to all media servers takes up lots of computing resources.

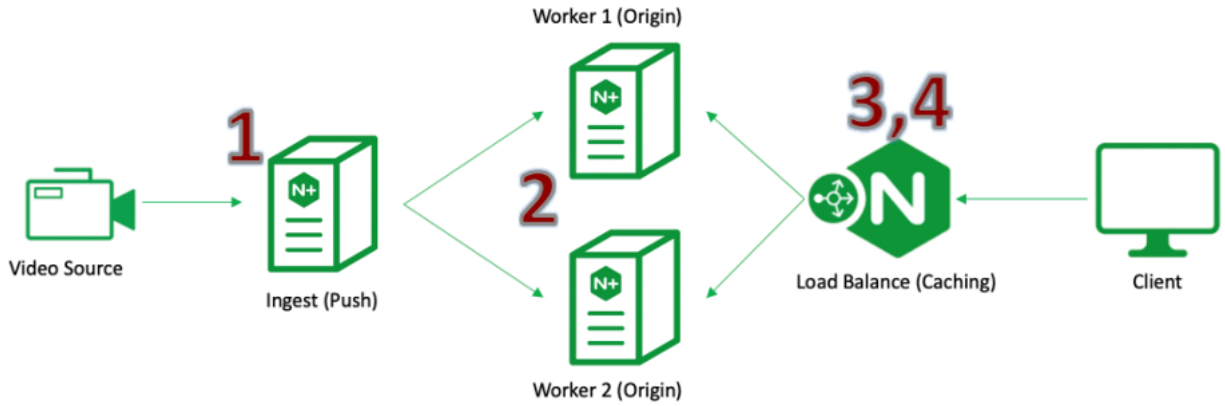
At the distribution sites (Client)

- Another transcoding job at media servers and URL redirection server – Duplicate process.
- No caching enabled - each client requests fragment every few seconds; overloading media servers.

Looking at current config, there were duplicate transcoding job which were replaced with a more efficient design of Origin to Worker Nodes concept – similar to a Content Distribution Network. Also, customer created their own load-balancing algorithm with URL redirection nginx configurations. We have replaced that with Caching concepts instead. Remember, distributing traffic with DNS without knowing the actual health status isn't helpful.

Last edited 25 March 2022.

Enhanced Streaming Distribution Design with NGINX Plus



1. Offload transcoding operations from Ingest servers to 'Worker Nodes/Origin'
2. "Worker Nodes" act as a pool of scalable Origin servers to serve transcoded content (e.g. HLS, DASH)
3. At each site, define a Load Balancer (LB) that directs user to healthiest "Worker Node" for video sessions
4. At each site, on the same Load Balancer instance, add-on caching abilities so other users get video fragments faster from LB itself rather than retrieving/pulling fragments from "Worker Nodes" all the time.

Last edited 25 March 2022.

Benefits of NGINX Plus

Other than re-architecting the streaming flow, we want to solidify the value of NGINX Plus.

- **Official and certified NGINX Plus's RTMP module**
    - Performs faster and more optimized
        - This is a claim from NGINX team, we did not prove it. Perhaps no testing was officially done internally.
        - However, customer did two live streaming events and received very good feedback on user experience. This boosted customer's confidence.
- **Supports a Content Distribution Architecture ("CDN")**
    - Load-Balancing
        - Many methods to optimize performance. Take your pick.
    - Caching Capabilities
        - For HLS and DASH fragments. Less relevance to playlist as no need to cache older playlist for a live streaming event
    - Active health-checks to Origin Servers with customized parameters
        - Customized parameters are important as video streaming behaviour were beyond our knowledge at times.
    - High Availability Setup
        - Assurance that we can mitigate downtime
    - Dash and Adaptive Bitrate Support
        - Perhaps not the strongest argument as OSS may catch up
        - Also, customer chose HLS format instead of DASH. HLS supported by OSS too.
- **Richer telemetry for streaming and user-experience metrics**
    - Customer was interested in NGINX dashboards. Another datapoint to monitor streaming.
    - Customer has adopted Prometheus/Grafana for ALL sites monitoring.

Last edited 25 March 2022.

Learnings

HLS Behaviour in a Worker Node Failover Scenario

There was a compromise from customer between using DASH or HLS. DASH has no issue during nginx worker failover and resumes. This means video keeps playing even if one Worker is down and then the down Worker resumes. On the other hand, HLS behaves differently in this Worker failover/resume scenario. This means video clients like VLC need to reload the stream otherwise a stream will be frozen. We found a forum talking about such VLC behaviour, it seems video player needs to be “smarter” which is beyond NGINX.

Despite the issue of HLS’s failover/resume behaviour, HLS provides an advantage over DASH in streaming static sprites/content like powerpoint. DASH could not refresh and gets hanged on one screen even though a person moved his mouse or changed to a next slide. This scenario is when a user pauses at one slide to talk through before changing slide.

Nobody investigated why this behaviour between DASH and HLS. But eventually customer chose HLS because of the importance of streaming static content.

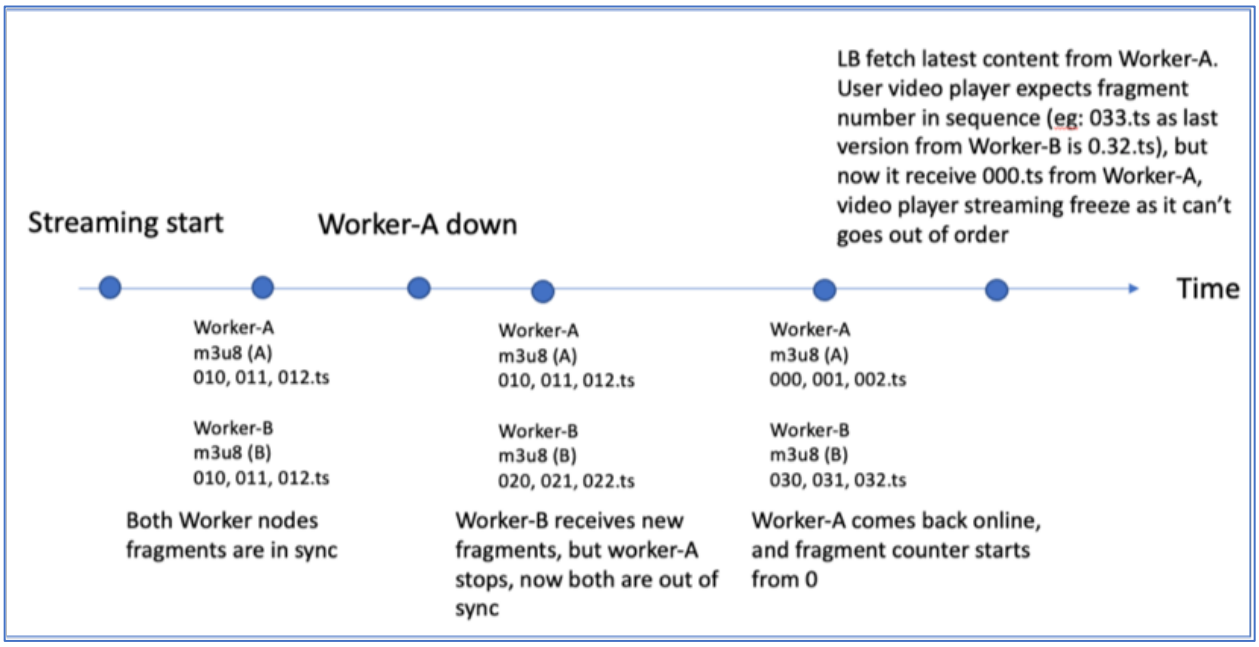
While HLS behaves non-ideally when there is a worker failover, we gave them a workaround to manually manipulate the worker nodes in an upstream pool. See next page.

Last edited 25 March 2022.

This is the HLS observation we managed to catch in our lab. In fact, someone else commented on this exact HLS behaviour too. “HLS media sequence resets make the player get stuck in buffering state” (https://github.com/google/ExoPlayer/issues/2872).

It looked like a bug but you know in the OSS world, it takes a long time to be fixed.

HLS Behaviour – Frames out of Sync/Frozen Frame after down instance UP again



There is a renaming of fragments from recovered worker. The worker must be thinking this is a new stream and is unaware it has recovered in the middle of a live stream.

Most video players (including VLC) aren’t smart enough to understand this concept too.

Meaning, although player downloaded the latest m3u8 from recovered worker but still insist on finding other non-existent higher-ordered fragments; trying to catch up where the last m3u8 should be.

Last edited 25 March 2022.

Possible operation improvement to address high availability

- There are 2 worker nodes, Primary worker node down and Backup node serves traffic. (So far so good here)
- Possible suggestion is to setup 3 workers nodes (Primary, Backup and Reserved).
- 3 of the workers nodes will constantly receive latest stream from Ingest node, their fragment files are in sync. (Meaning all renaming of fragments are same for ALL three workers)
- When Worker-A (Primary) is down, then only Worker-B (Backup) will take over and serve traffic to LB. Worker-C (Reserved) is currently marked Down and will not serve traffic to LB at all.
- After Worker-A goes offline, admin can update the LB configurations to mark Worker-B as primary and Worker-C as backup while removing the down flag on worker-C, reloading NGINX configs in NGINX LB will not have service impact.
    - See below config example

During ‘peace-time’ under your LB/Cache config:

```
upstream originworkers {
    zone originworkers 64k;
    least_conn;
    server 10.1.1.4; #primary
    server 10.1.1.8 backup; #backup
    #server 10.1.1.11 down; #reserve

}
```

Last edited 25 March 2022.

When primary node fails and you want to prevent it from serving even if it is up:

```
upstream originworkers {

    zone originworkers 64k;
    least_conn;
    server 10.1.1.4 down; #primary becomes down to prevent recovery
    server 10.1.1.8; #Backup becomes Primary. Remove the flag 'backup'
    server 10.1.1.11 backup; #reserve becomes backup. Replace "down" with "backup.

  }
```

You can manually replace the configuration and nginx -s reload. NGINX reload does not impact services.

In your current config, it takes 50 seconds to UP a recovered worker. In real scenario, if a worker is down (very rare nginx daemon will be down) and is UP, it will take 50 seconds. (Interval=5, passes=10).

Details of fragments out-of-sync observed

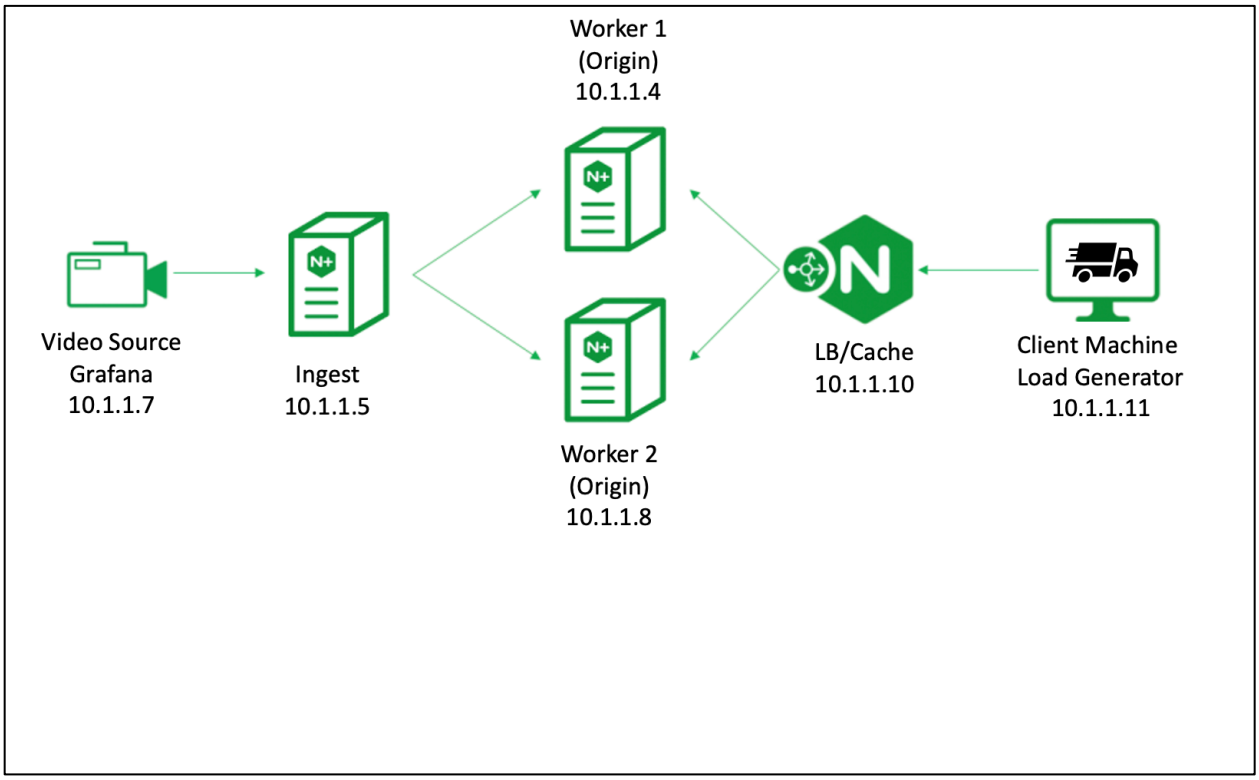
The video player isn't smart enough to only read the latest m3u8 file from recovered worker node. It continues to find m3u8 manifest that contains fragments that it did not receive since its last fragment. Because this is a live stream, these fragments are not available anymore.

Example. Worker 1 fragments 0 1 2 3 and its m3u8 is played by video player. When Worker 1 is down, player continues to receive fragments 4 5 6 7 and its m3u8 and no issue with this. When Worker 1 is recovered, the hls fragments are renamed as 0 1 2 3 with its m3u8. In video content this is not wrong as Ingest server is still breaking up the fragments correctly. However, Worker 1 who has just recovered thought it was a stream and it has no 'memory' of the previous stream, hence Worker 1 names fragments as 0 1 2 3 again.

The video player when receiving Worker 1's fragments 0 1 2 3 and m3u8 list find it strange and thought it was out-of-sync. Video player tries to be smart and find fragments 4 5 6 7 from Worker 1 but Worker 1 does not have yet. 1234 to 4567 seems like a "next-jump" but if worker 1 takes minutes to be UP, the jump could be 50-100 fragments apart. In essence, a smart video player needs to be built or bought but requires customer's effort. (In this case, we used a workaround to make sure down NGINX instance will NOT be UP again)

Last edited 25 March 2022.

Our Testing Environment



- Video source broadcasts to Ingest Server (NGINX Plus with RTMP Module).
- Ingest Server will push video source to Origin servers (NGINX Plus with RTMP Module).
- Origin Servers will transcode video into HLS/DASH fragments as well as provide m3u8/mpd playlist.
- Load Balancers/Cache (NGINX Plus) will point to Worker 1 and 2 as Upstream. Caching of fragments meant each site's video client can get fragments quickly from Cache rather than to retrieve from Workers.

Start Video Broadcast (10 minutes video)

Video: https://test-videos.co.uk/bigbuckbunny/mp4-h264

start broadcasting with *sh stream.sh*

Stream script

ffmpeg -re -i bbb_sunflower_1080p_60fps_normal.mp4 -vcodec copy -loop -1 -c:a aac -b:a 160k -ar 44100 -strict -2 -f dash -f flv rtmp://10.1.1.5/live/bbb

Last edited 25 March 2022.

At Worker 1 and 2, you will see the fragments being populated.

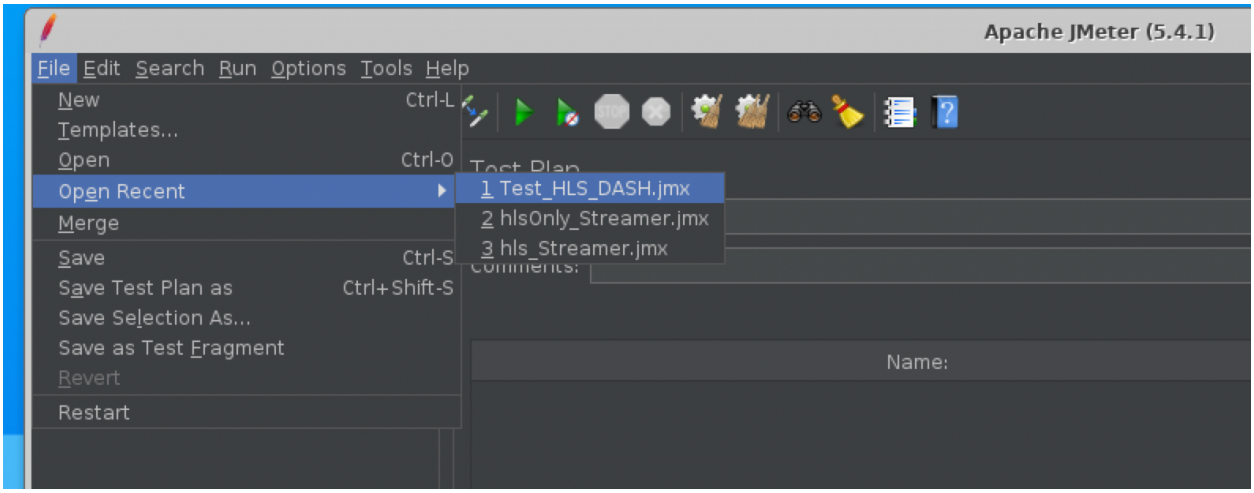
```
[root@worker1 tmp]# ls
dash  hls  nginx-rtmp.0  nginx-rtmp.1  nginx-rtmp.2  nginx-rtmp.3  systemd-private-128c4ac0b3de4daea27ce994015b46df-chronyd.service-FxwiUT
[root@worker1 tmp]# ls dash
bbb-16256.m4a  bbb-23.m4a  bbb-24290.m4a  bbb-31590.m4a  bbb-39723.m4a  bbb-48056.m4a  bbb-8340.m4a  bbb-init.m4a  bbb.mpd      bbb-raw.m4v
bbb-16256.m4v  bbb-23.m4v  bbb-24290.m4v  bbb-31590.m4v  bbb-39723.m4v  bbb-48056.m4v  bbb-8340.m4v  bbb-init.m4v  bbb-raw.m4a
[root@worker1 tmp]# ls hls
bbb-0.ts  bbb-1.ts  bbb-2.ts  bbb-3.ts  bbb-4.ts  bbb-5.ts  bbb-6.ts  bbb-7.ts  bbb.m3u8
```

/tmp/dash and /tmp/hls of Worker 1 and 2.

Launch Video Client

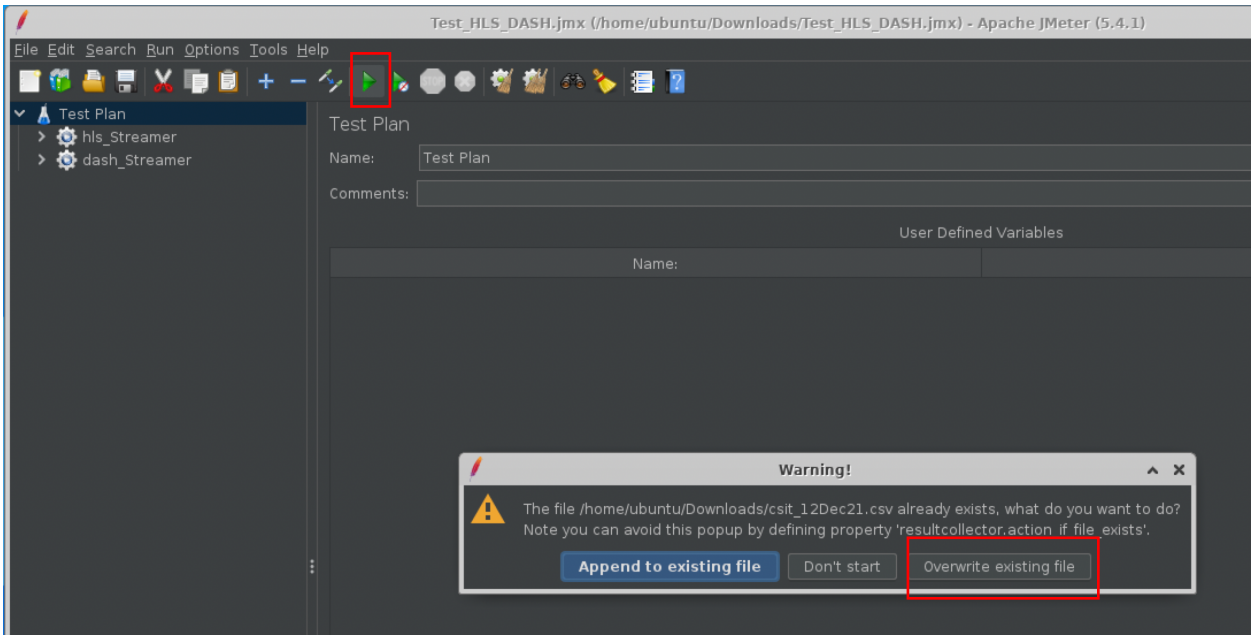
Launch jMeter

Note: I am using a MAC's Remote Desktop Connection and sometimes XRDP window cannot capture your mouse clicks properly. Simply close and relaunch XRDP.



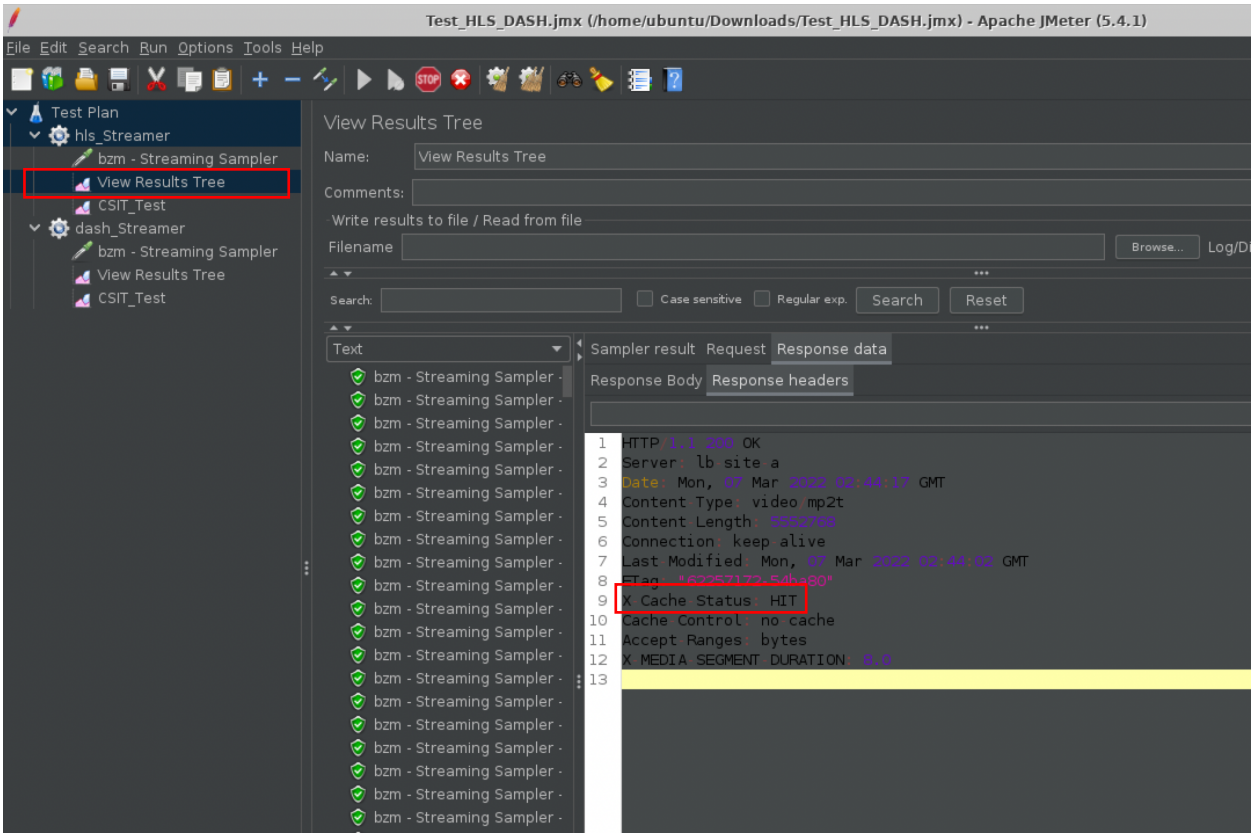
Load Test_HLS_DASH.jmx

Last edited 25 March 2022.



Click Start Icon. Overwrite existing file. It was meant to capture raw data into CSV file, unimportant.

Prove caching from JMeter Test

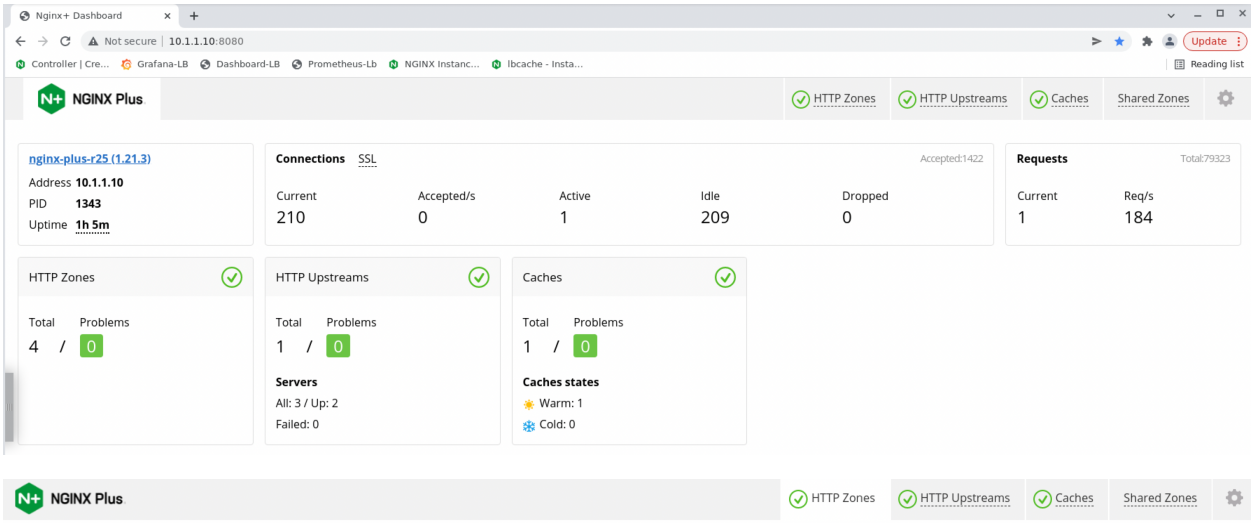


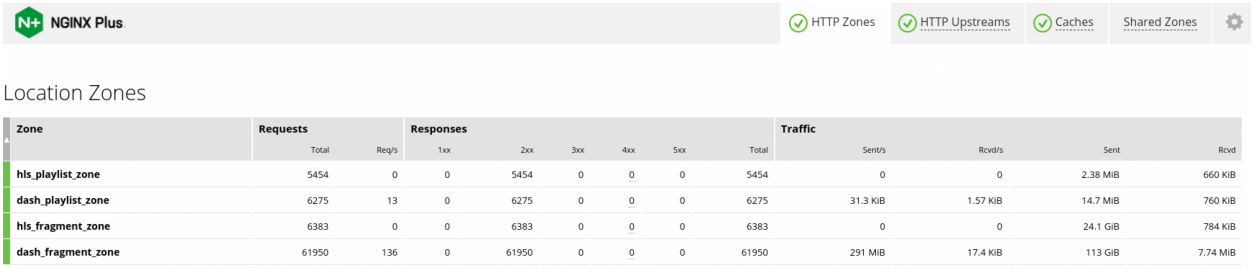
Last edited 25 March 2022.

Click on hls_Streamer or dash_Streamer to show fragments being consumed. Click on View Results Tree. Select a result, select Response data, select Response headers to show Caching evidence.

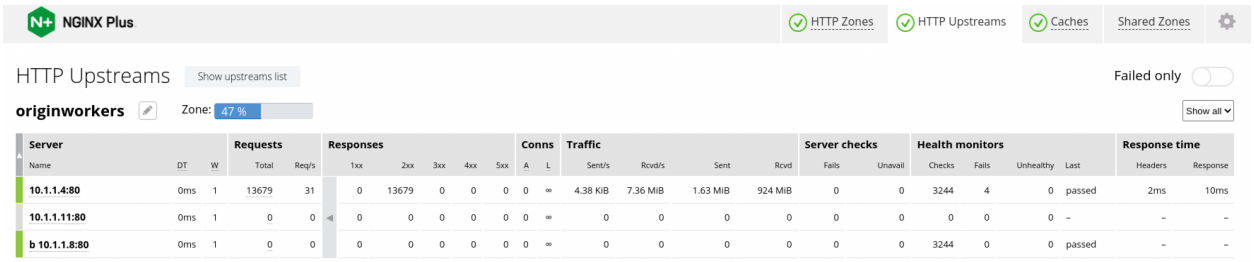
Show NGINX Plus Dashboard

Launch Web Browser and browse to your LB/Cache instance's nginx dashboard



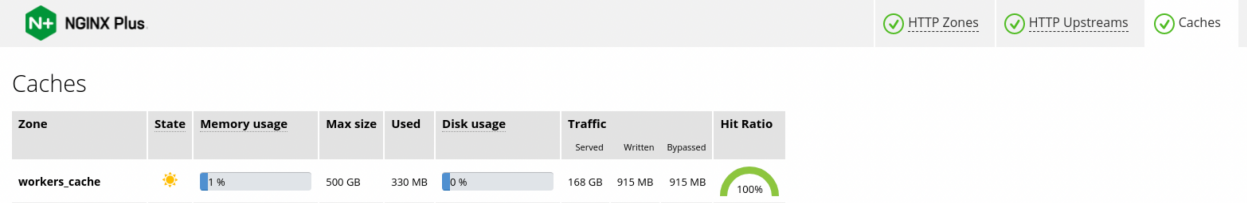


Show there are no Errors. No 4xx.



Upstream Status. We will do a failover test in next section. Notice 10.1.1.4 is Primary, 10.1.1.11 is "reserve", 10.1.1.8 is Backup.

Last edited 25 March 2022.



Show Cache status

Simulate Failover

Due to the behaviour of HLS, a recovered Origin Worker will cause the video client to stop playing fragments. See this document's POC Learnings – "HLS Behaviour in a Worker Node Failover Scenario"

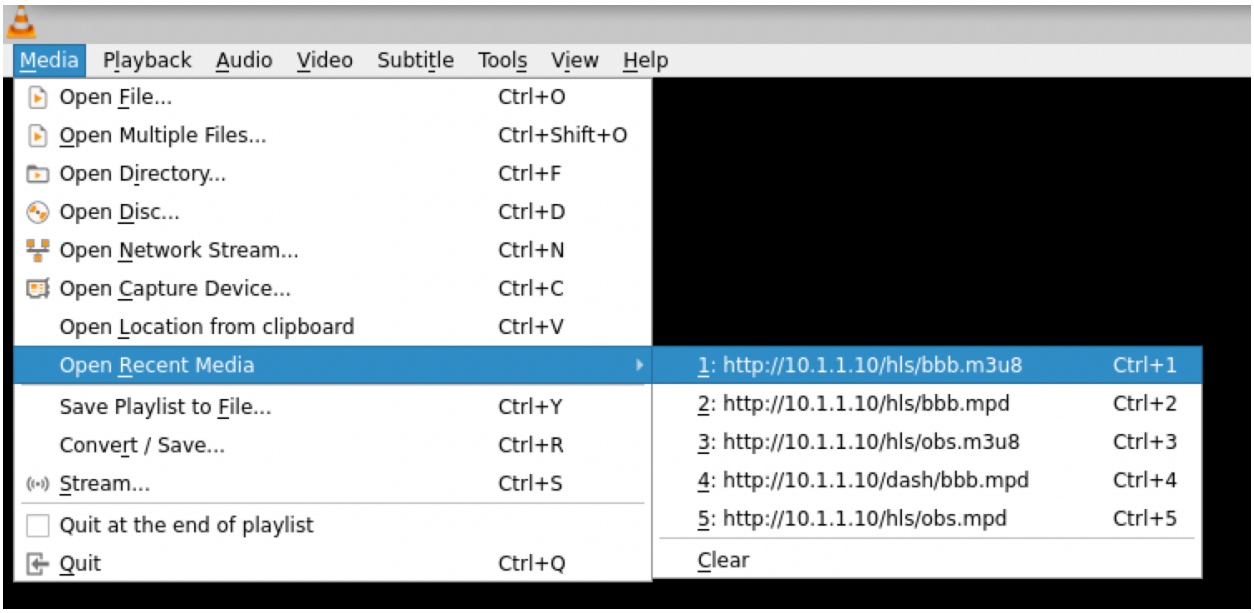
DASH format will not have this issue described above. However, in displaying powerpoint or static sprites, DASH seem to be an issue where it does not refresh static content.

start broadcasting with *sh stream.sh*

ffmpeg -re -i bbb_sunflower_1080p_60fps_normal.mp4 -vcodec copy -loop -1 -c:a aac -b:a 160k -ar 44100 -strict -2 -f dash -f flv rtmp://10.1.1.5/live/bbb

Click on Application Finder and Launch VLC Media Player

Last edited 25 March 2022.



Launch m3u8 playlist or mpd playlist. Do not use this screenshot for the path of playlist, look at your nginx config where these playlist are published.

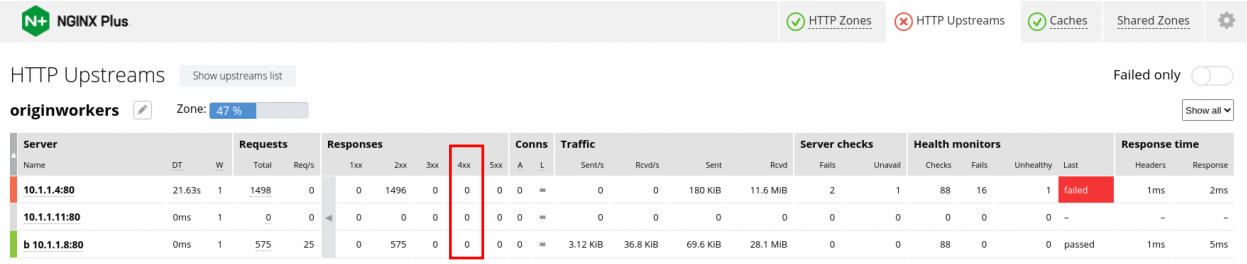
You may also want to launch JMeter too and prove there are no failures from NGINX Dashboard.


Video Plays

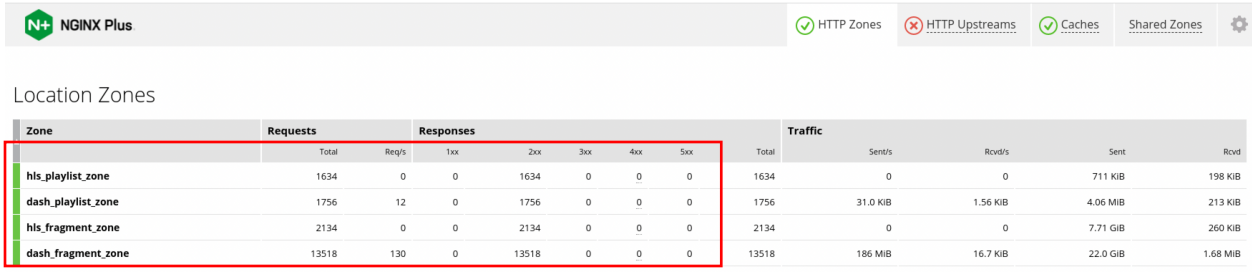
Last edited 25 March 2022.

Stop nginx process in Worker 1 to simulate a failure

SSH to Worker 1 and *systemctl stop nginx*



Show video playback in VLC player has no impact. NGINX dashboard shows zero 4xx errors.



Another viewpoint where both playlist and fragments has no 4xx errors.

If you want to systemctl start nginx in worker 1, remember the issue as described in this document's POC Learnings - *HLS Behaviour in a Worker Node Failover Scenario*

It is interesting to note that from NGINX Dashboard, no errors were observed when Worker 1 is resumed. But in video playing perspective, the video was frozen. This is a video player's design?

The only time we see 4xx errors is when JMeter or VLC still tries to retrieve playlist when a stream is over.

Last edited 25 March 2022.

NGINX Plus Configurations

Ingest Server

```
load_module modules/ngx_rtmp_module.so;
load_module modules/ngx_http_js_module.so;

worker_processes  auto;

events {
    worker_connections  1024;
}

error_log /var/log/nginx/error.log;

rtmp_auto_push on;

rtmp {
    server {
        access_log /var/log/nginx/access.log;

        listen 1935;
        listen 1935;
        chunk_size 4000;
        #buflen 15s;

        application live {

          live on;
          interleave on;

          push 10.1.1.4; #worker node
          push 10.1.1.8; #worker node

          deny play all;

        }
    }
}


http {

    server {

        listen 80;

        # This URL provides RTMP statistics in XML
        location /stat {
            rtmp_stat all;

            # Use this stylesheet to view XML as web page
            # in browser
```

```
            rtmp_stat_stylesheet stat.xsl;
        }

        location /stat.xsl {
            # XML stylesheet to view RTMP stats.
            # Copy stat.xsl wherever you want
            # and put the full directory path here
            root /etc/nginx/stat.xsl;
        }

    }

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;
  include /etc/nginx/conf.d/*.conf;
}
```

Last edited 25 March 2022.

Worker Nodes (Same for all Nodes)

```
load_module modules/ngx_rtmp_module.so;
load_module modules/ngx_http_js_module.so;


worker_processes  auto;

events {
    worker_connections  1024;
}

error_log /var/log/nginx/error.log;

rtmp_auto_push on;

rtmp {
    server {
        access_log /var/log/nginx/access.log;

        listen 1935;
        chunk_size 4000;

        application live {

          live on;
          interleave on; #optimization of combining video and audio chunks in
single RTMP packet

          hls on; #for hls playback
          hls_path /tmp/hls; #hls fragments store
          hls_fragment 6s;
          hls_playlist_length 30;
          hls_continuous on; #newly added. Eliminates 4xx errors so far.

          dash on;
          dash_path /tmp/dash;
          dash_fragment 6s;
          dash_playlist_length 30;

          allow publish 10.1.1.5; #Ingest server
          deny publish all;

        }
    }
}

http {

default_type application/octet-stream; #added 14Dec21

    server {

        access_log off;
        listen 80;
```

```
        #access_log /var/log/nginx/access.log;

        types{
            application/vnd.apple.mpegurl m3u8; #HLS meta-data
            video/mp2t ts; #HLS fragments
            text/html html;
            application/dash+xml mpd; #DASH meta-data
        }

        #used to pass load balancer health check
        location = / {
           return 200;
        }

        # This URL provides RTMP statistics in XML
        location /stat {
            rtmp_stat all;

            # Use this stylesheet to view XML as web page
            # in browser
            rtmp_stat_stylesheet stat.xsl;
        }

        location /stat.xsl {
            # XML stylesheet to view RTMP stats.
            # Copy stat.xsl wherever you want
            # and put the full directory path here
            root /etc/nginx/stat.xsl;
        }

        location /download { #Ignore this part. It's my experiment.
         types {
            application/octet-stream bin exe dll;
          }
        }

        location /hls {

           access_log /var/log/nginx/access.log;
            # Serve HLS fragments
            root /tmp;
        }

        location /dash {

           access_log /var/log/nginx/access.log;
            # Serve DASH fragments
            root /tmp;

        }

    }

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;
  include /etc/nginx/conf.d/*.conf;
}
```

Last edited 25 March 2022.

LB/Cache Nodes ("Peace" Time)

The settings were taken from customer's OSS current settings as well as from an nginx performance test done.

```
load_module modules/ngx_http_js_module.so;

worker_processes  auto;
worker_rlimit_nofile 100000;

events {
    worker_connections  4096;
    accept_mutex off;
    multi_accept on;
}
error_log /var/log/nginx/error.log;

http {
    aio on;
    sendfile on;
    directio 512;
    keepalive_timeout 65; #May not be useful for video stream, but best practice.
    access_log off;
    proxy_cache_path /tmp/cache_storage levels=1:2 keys_zone=workers_cache:10m inactive=5m max_size=500G;
    keyval_zone              zone=workers_files_zone:1m              type=string state=/tmp/workers_files_zone.keyval;
    map $request_method $purge_method {
        PURGE 1;
        default 0;
    }


    match health_conditions {
       status 200;
       #header Content-Type = application/octet-stream; #CURL and you know Worker returns stream, not HTML!
    }

    upstream originworkers {
        zone originworkers 64k;
        least_conn;
        server 10.1.1.4; #primary at peacetime
        server 10.1.1.8 backup; #backup at peacetime
        server 10.1.1.11 down; #reserve at peacetime

    }

    server {

        listen 80;
        server_tokens "lb-site-a";
        add_header X-Proxy-Cache $upstream_cache_status;
```

```
        location / {
          index index.html; #each event is a new page

          root /etc/nginx/page ;

        }

        location ~ /hls/.+\.m3u8$ {

            status_zone hls_playlist_zone;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

        }

        location ~ /dash/.+\.mpd$ {

            status_zone dash_playlist_zone;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

        }


        location /hls {
            status_zone hls_fragment_zone;
            #proxy_http_version 1.1; #This stops VLC from playing.
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

            proxy_cache workers_cache;
            proxy_cache_key $scheme$host$request_uri;
            proxy_cache_valid 200 301 302 5m;
            #proxy_cache_valid any 1m;
            add_header X-Cache-Status $upstream_cache_status;
            add_header Cache-Control no-cache;

            proxy_cache_purge $purge_method;
        }

        location /dash {
            status_zone dash_fragment_zone;
            #proxy_http_version 1.1; #This stops VLC from playing.
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;
```

```
            proxy_cache workers_cache;
            proxy_cache_key $scheme$host$request_uri;
            proxy_cache_valid 200 301 302 5m;
            #proxy_cache_valid any 1m;
            add_header X-Cache-Status $upstream_cache_status;

            proxy_cache_purge $purge_method;
        }

    }

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;
  include /etc/nginx/conf.d/*.conf;
}
```

Last edited 25 March 2022.

LB/Cache Nodes ("Failover" Time)

```
load_module modules/ngx_http_js_module.so;

worker_processes  auto;
worker_rlimit_nofile 100000;

events {
    worker_connections  4096;
    accept_mutex off;
    multi_accept on;
}
error_log /var/log/nginx/error.log;

http {
    aio on; #Perf Test
    sendfile on;
    directio 512;
    keepalive_timeout 65; #May not be useful for video stream, but best
practice.
    access_log off;
    proxy_cache_path /tmp/cache_storage levels=1:2 keys_zone=workers_cache:10m
inactive=5m max_size=500G;
    keyval_zone              zone=workers_files_zone:1m              type=string
state=/tmp/workers_files_zone.keyval;
    map $request_method $purge_method {
        PURGE 1;
        default 0;
    }


    match health_conditions {
       status 200;
       #header Content-Type = application/octet-stream; #CURL and you know
Worker returns stream, not HTML!
    }

    upstream originworkers {
        zone originworkers 64k;
        least_conn;
        server 10.1.1.4 down; #prevents recovery
        server 10.1.1.8; #becomes primary
        server 10.1.1.11 backup; #becomes backup

    }

    server {

        listen 80;
        add_header X-Proxy-Cache $upstream_cache_status;

        location / {
          index index.html;

          root /etc/nginx/page ;
```

```
        }

        location ~ /hls/.+\.m3u8$ {

            status_zone hls_playlist_zone;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

        }

        location ~ /dash/.+\.mpd$ {

            status_zone dash_playlist_zone;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

        }


        location /hls {
            status_zone hls_fragment_zone;
            #proxy_http_version 1.1; #This stops VLC from playing.
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

            proxy_cache workers_cache;
            proxy_cache_key $scheme$host$request_uri;
            proxy_cache_valid 200 301 302 5m;
            #proxy_cache_valid any 1m;
            add_header X-Cache-Status $upstream_cache_status;
            add_header Cache-Control no-cache;

            proxy_cache_purge $purge_method;
        }

        location /dash {
            status_zone dash_fragment_zone;
            #proxy_http_version 1.1; #This stops VLC from playing.
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass http://originworkers;
            health_check match=health_conditions interval=5 fails=3 passes=10;

            proxy_cache workers_cache;
            proxy_cache_key $scheme$host$request_uri;
            proxy_cache_valid 200 301 302 5m;
```

```
            #proxy_cache_valid any 1m;
            add_header X-Cache-Status $upstream_cache_status;

            proxy_cache_purge $purge_method;
        }

    }

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;
  include /etc/nginx/conf.d/*.conf;
}
```

Last edited 25 March 2022.

JMeter Setup – Load Testing, Concurrent Users/Threads Test

Java error when you tried launching jMeter

After unzipping (sudo apt-get install zip unizip) the downloaded apache-jmeter-5.4.1 file. You navigate to cd Downloads/apache-jmeter-5.4.1/bin

And when you ./jmeter it gives Java home error. You need to install JRE like seen below:

sudo apt update

sudo apt install default-jre

```
ubuntu@ubuntu:~/Downloads/apache-jmeter-5.4.1/bin$ ./jmeter
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
At least one of these environment variable is needed to run this program
ubuntu@ubuntu:~/Downloads/apache-jmeter-5.4.1/bin$ java

Command 'java' not found, but can be installed with:

sudo apt install default-jre              # version 2:1.11-72, or
sudo apt install openjdk-11-jre-headless  # version 11.0.10+9-0ubuntu1~20.04
sudo apt install openjdk-13-jre-headless  # version 13.0.4+8-1~20.04
sudo apt install openjdk-14-jre-headless  # version 14.0.2+12-1~20.04
sudo apt install openjdk-8-jre-headless   # version 8u282-b08-0ubuntu1~20.04
```

JMeter Plugin (BlazeMeter HLS tester)

Overall Guide here: https://www.blazemeter.com/blog/HLS-3.0-Release

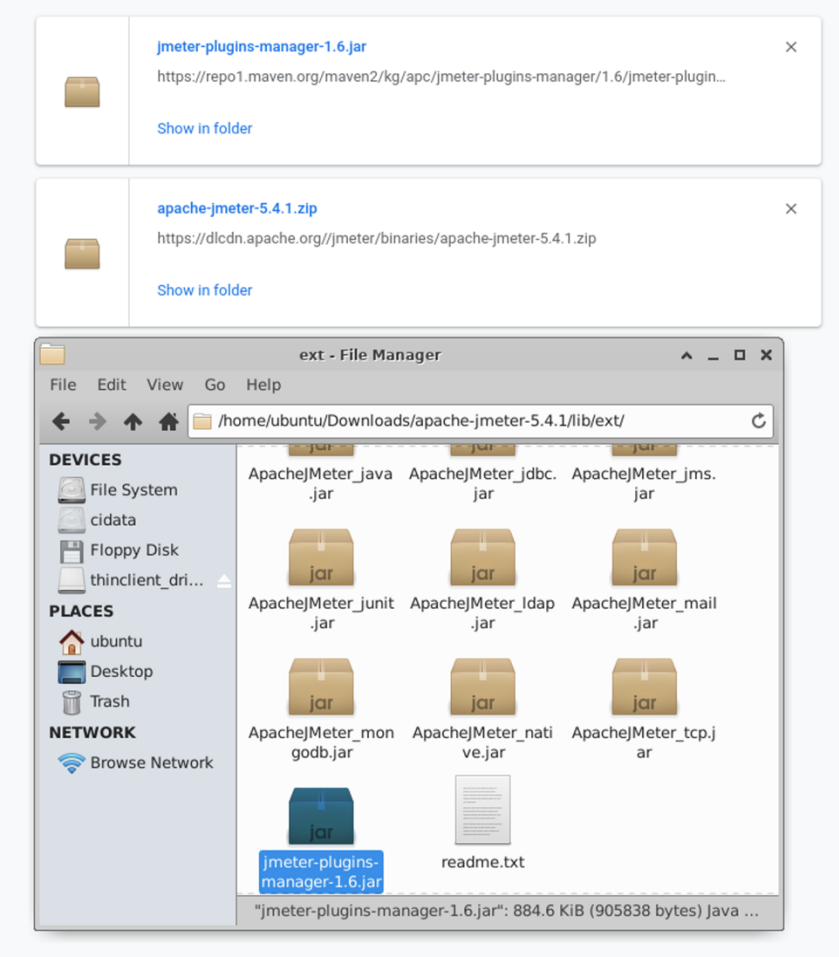
Pre-Requisites: jMeter Plugin Manager and BlazeMeter HLS Plugin

https://www.blazemeter.com/blog/how-install-jmeter-plugins-manager
(Follow steps from link or see my summary steps below)

Pre-Req Step 1: Download plugins-manager.jar and put it into lib/ext directory, then restart JMeter.

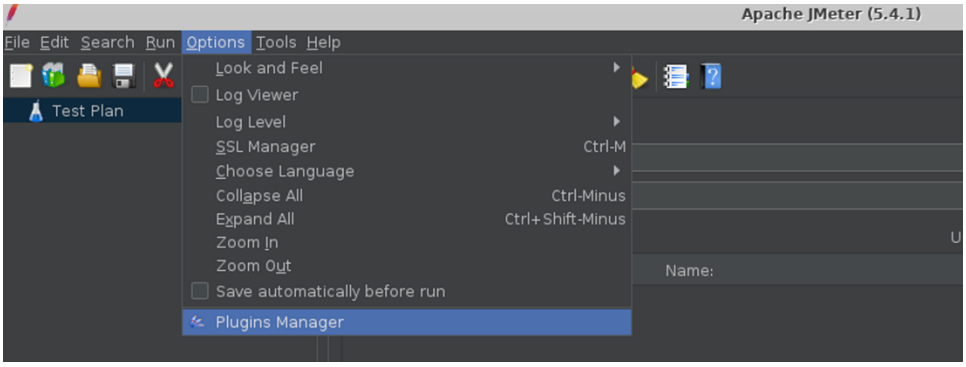
https://jmeter-plugins.org/install/Install/

Last edited 25 March 2022.



Restart jMeter if you have it launched before.

Pre-Req Step 2: Launch JMeter -> Options -> Plugins Manager

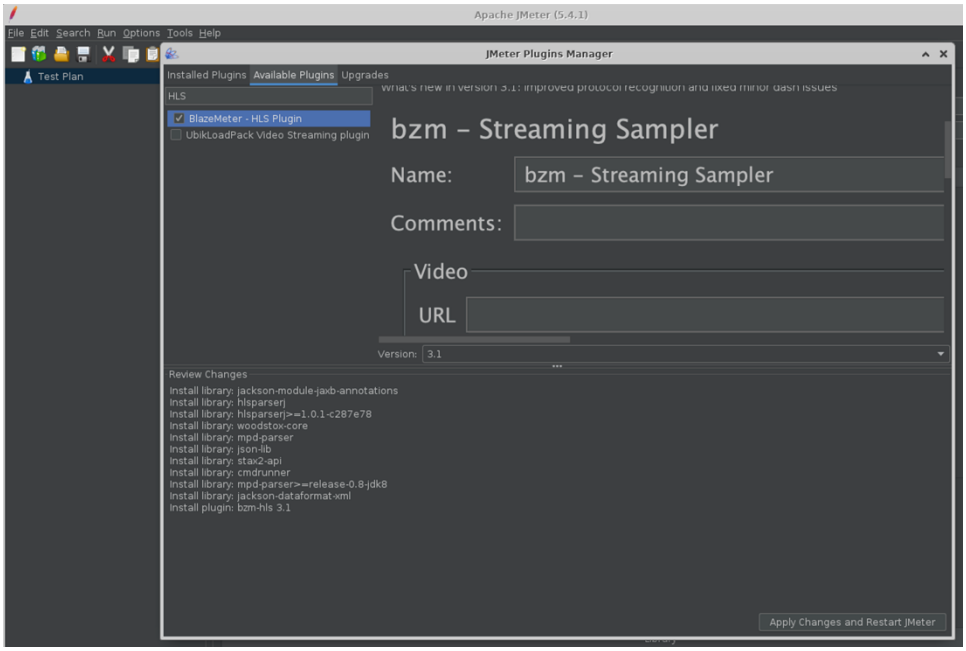


Pre-Req Step 3: Install BlazeMeter HLS Plugin (Search HLS under Available Plugins)

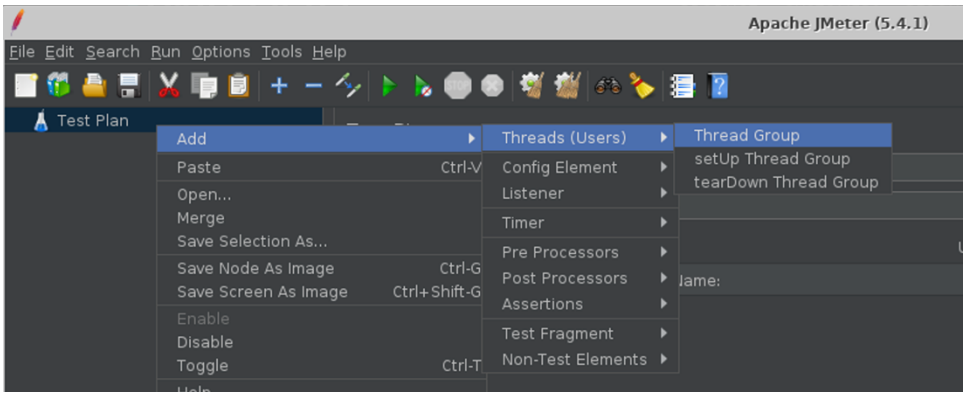
Apply Changes and Restart jMeter

Last edited 25 March 2022.

Bottom of the Plugin screen, you will see downloading messages



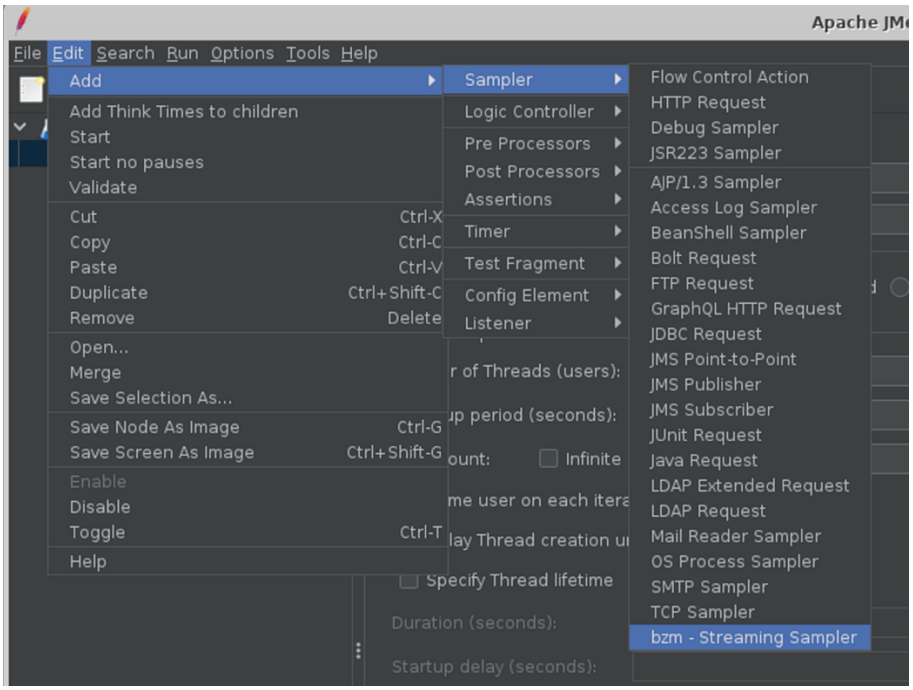
Set up the TEST



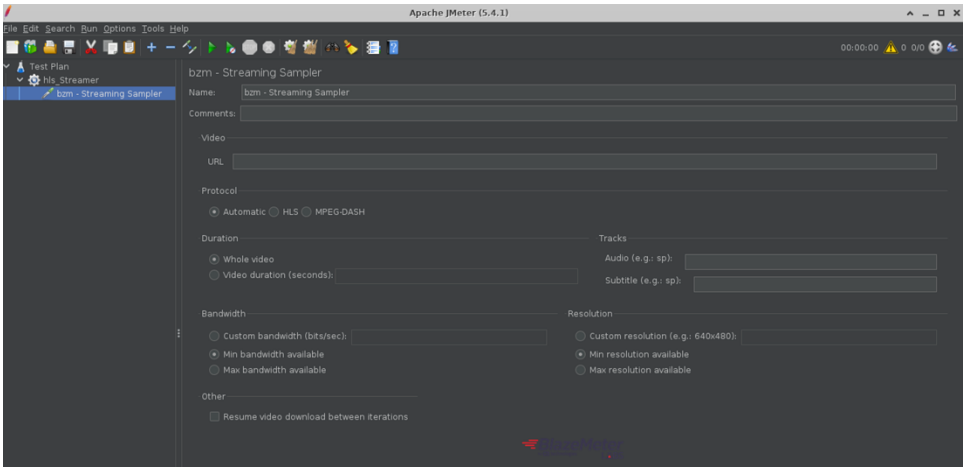
Name the elements of this thread group which represents your users. I named threadgroup as hls_streamer.

While your Test Plan -> hls_streamer is still highlighted, on Menu Bar -> Edit -> Add -> Sampler -> bzm –> Streaming Sampler

Last edited 25 March 2022.

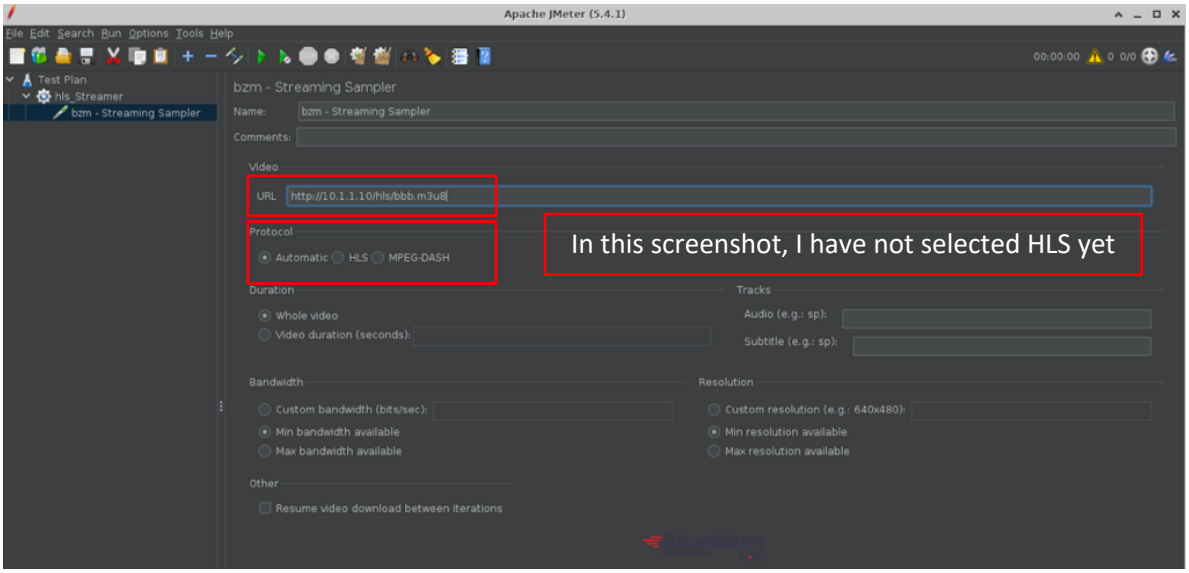


Below your thread group hls_Streamer you will see bzm – Streaming Sampler. I did not rename it so I know this is from BZM.

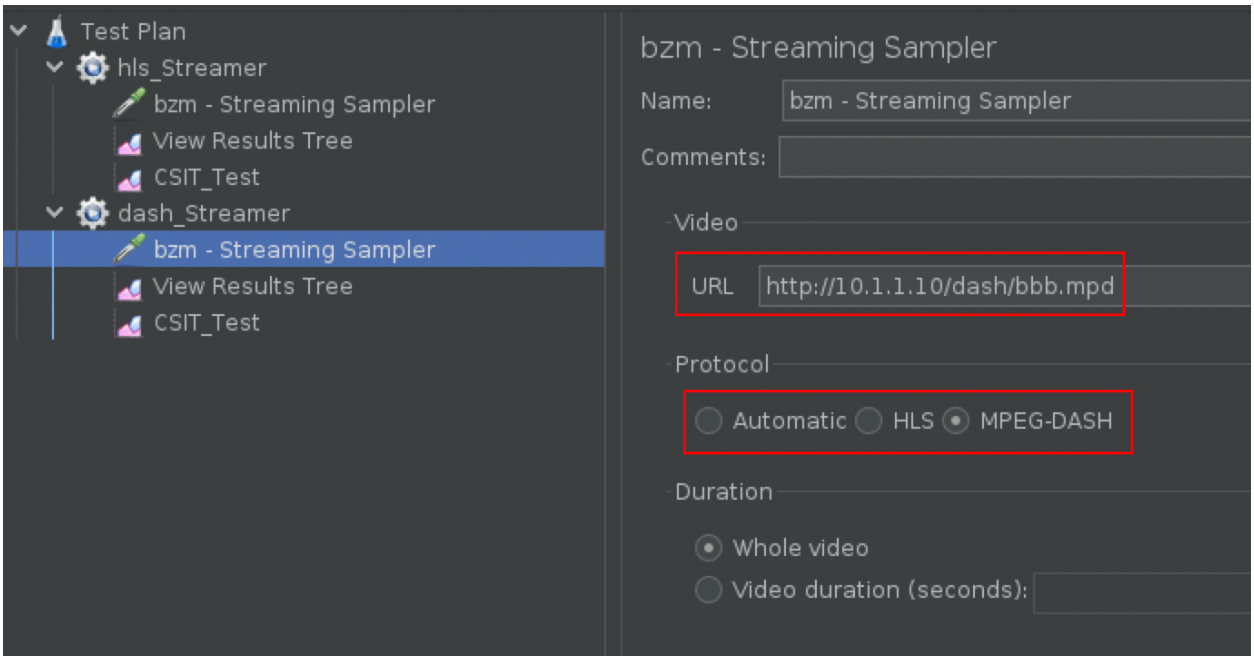


Last edited 25 March 2022.

Add in the URL of your m3u8 (Your LB/Cache)

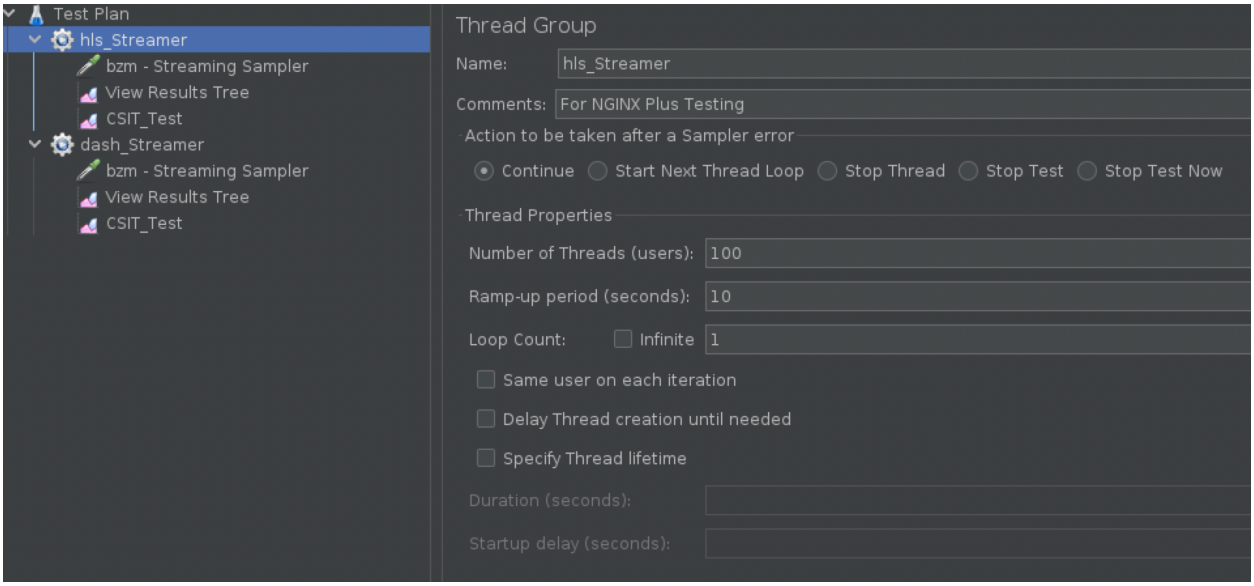


In this case, the m3u8 and fragments are published into the Worker nodes' HLS location.



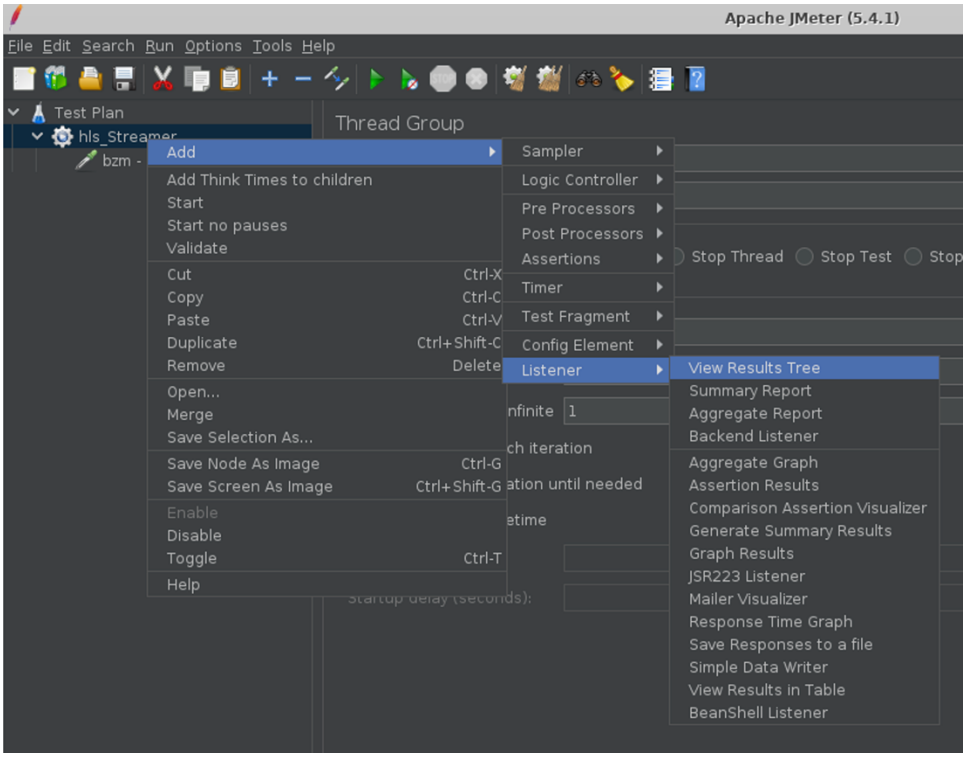
Last edited 25 March 2022.

Check your test plan Thread properties



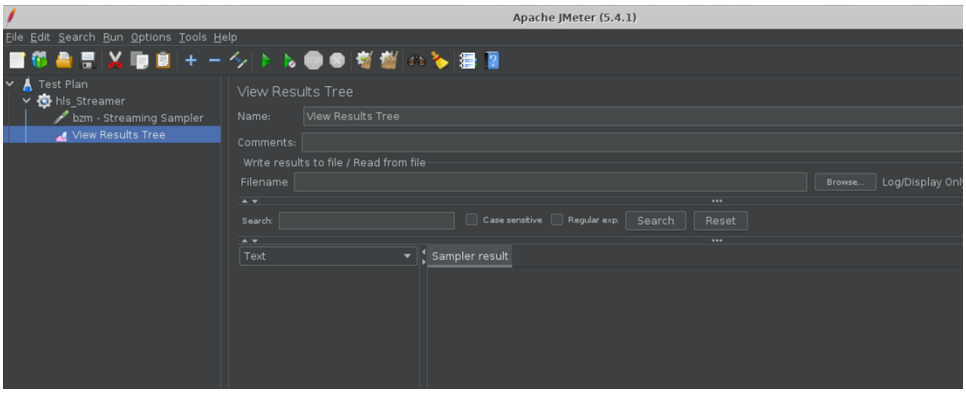
For testing, I separated HLS and DASH. This means worker nodes and LB/Cache are tested with 200 concurrent users.

Viewing the Results:



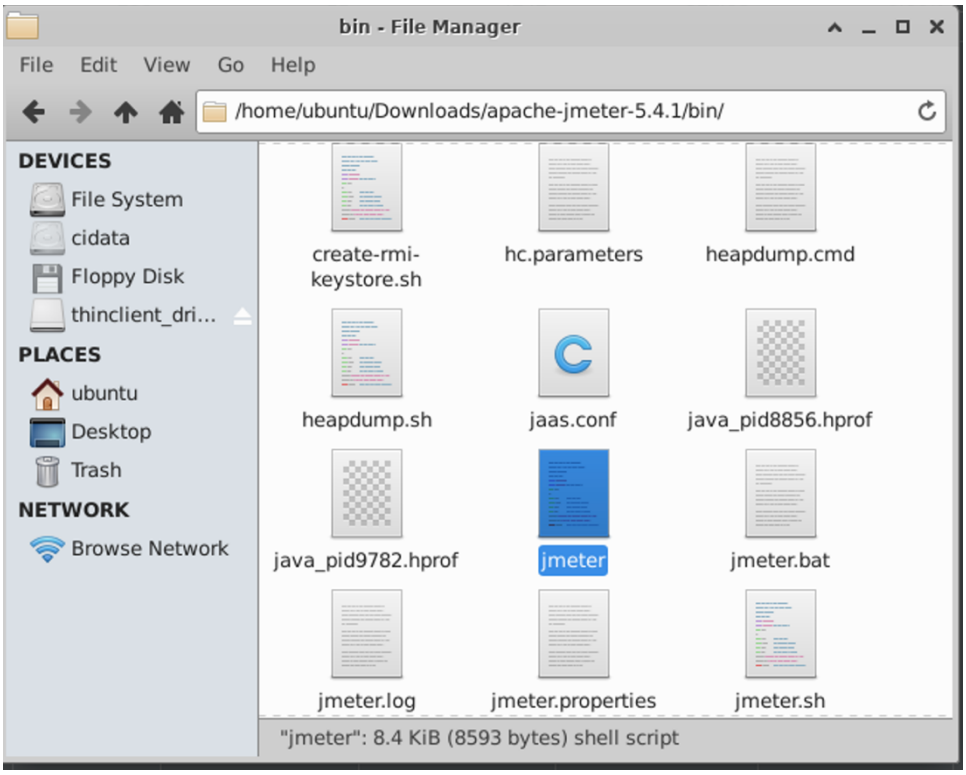
Once clicked, a new panel appears below hls_Streamer threadgroup.

Last edited 25 March 2022.



Solve the java.lang.OutOfMemoryError from jMeter

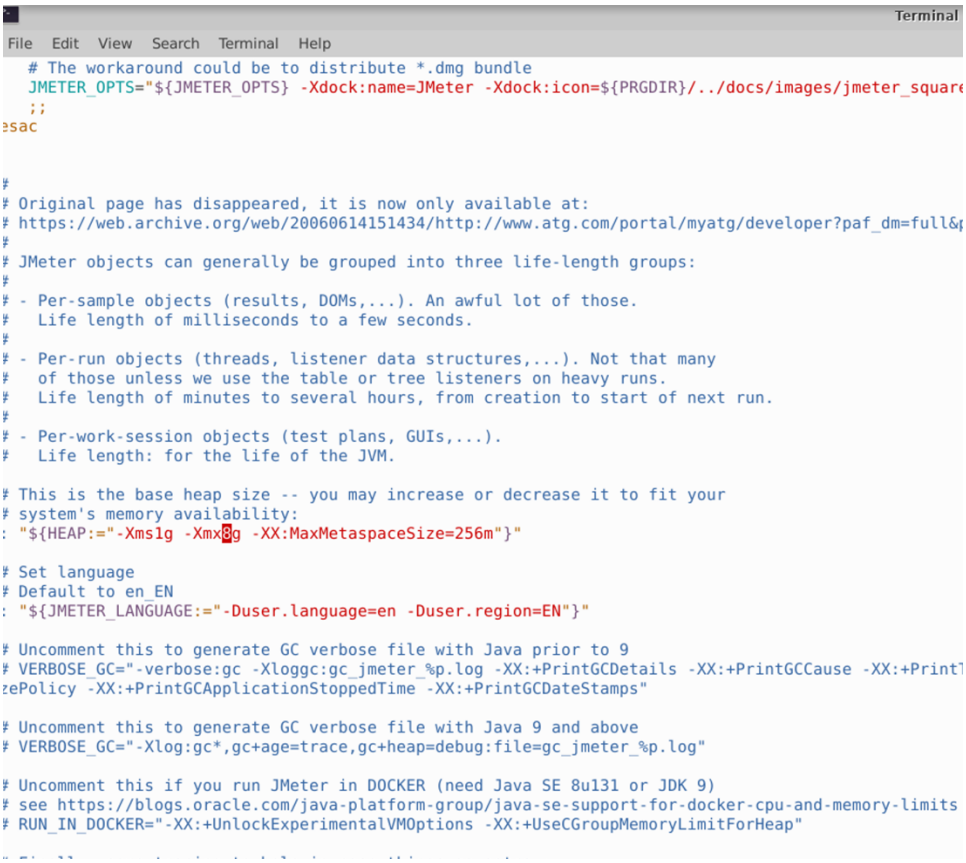
https://www.blazemeter.com/blog/9-easy-solutions-jmeter-load-test-%E2%80%9Cout-memory%E2%80%9D-failure

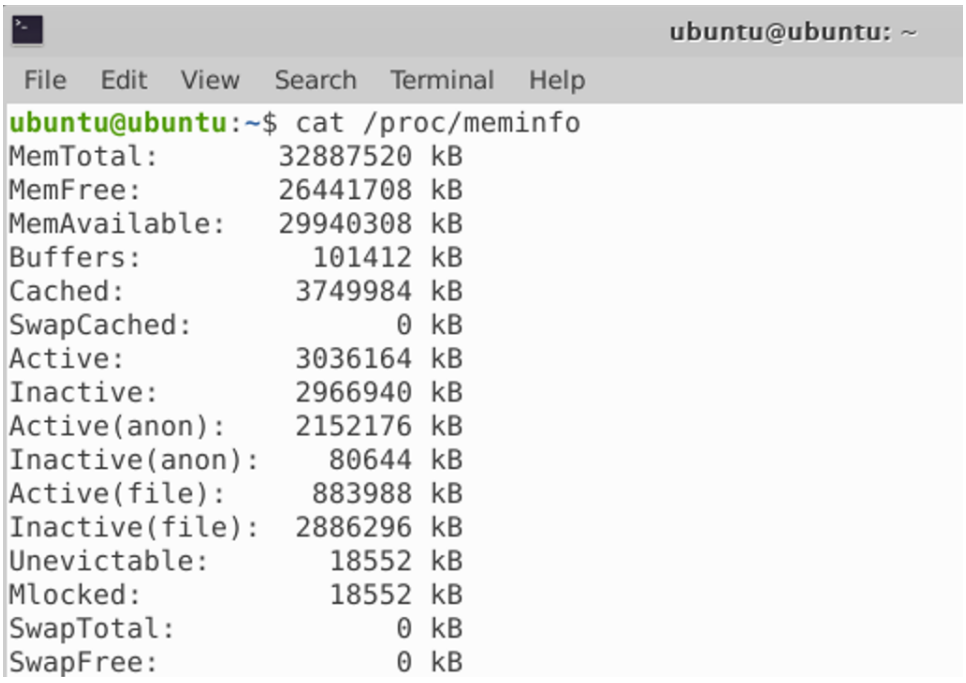


Edit jmeter file (Not jmeter.bat unless Windows OS), navigate to HEAP, change to Xms1g –Xmx24g

The screenshot I put 8 and adjust to 24. Indeed, 24GB helps but still cannot hit 100 users so you have to delay spawn rate.

Last edited 25 March 2022.



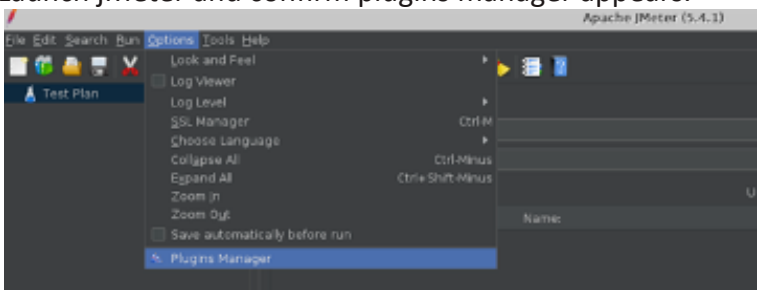


Last edited 25 March 2022.

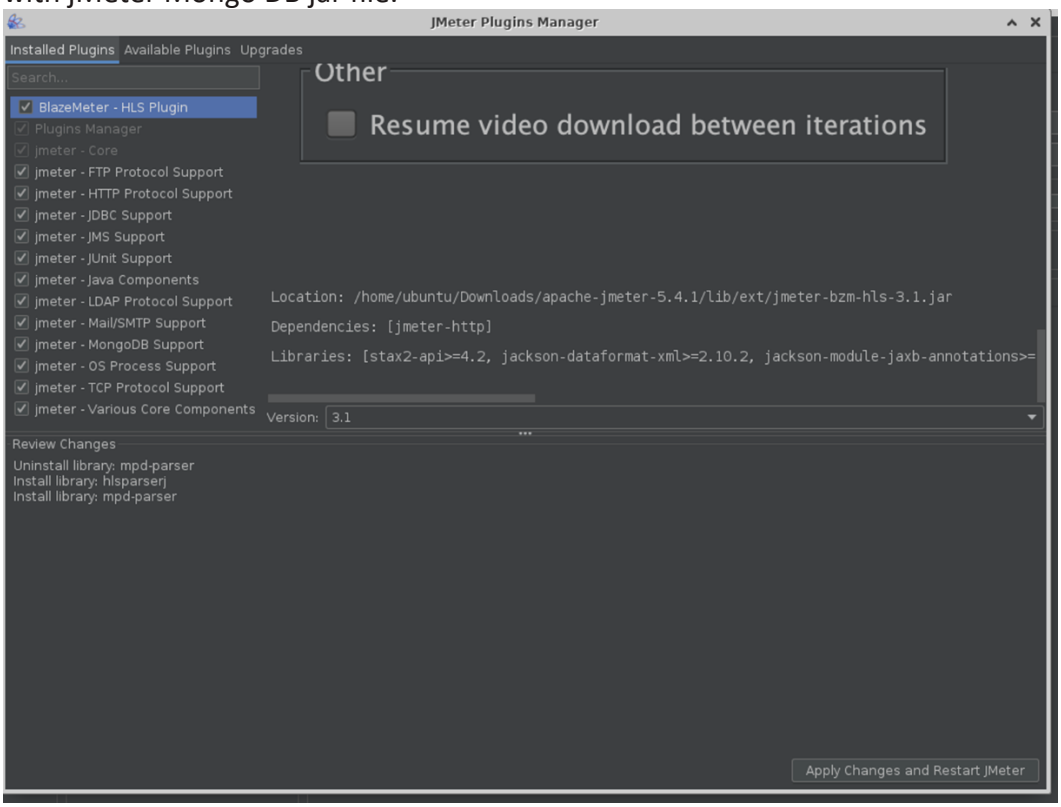
In a 32GB RAM VM, I have 29GB MemAvailable.
(https://superuser.com/questions/980820/what-is-the-difference-between-memfree-and-memavailable-in-proc-meminfo) I still don't know the difference, best use lower number.

Suggested steps to "install" jMeter Plugin for offline environments

Ensure jMeter is not launched.
Place jmeter-plugins-manager-1.6.jar into apache-jmeterFolder/lib/ext
Launch jMeter and confirm plugins manager appears.

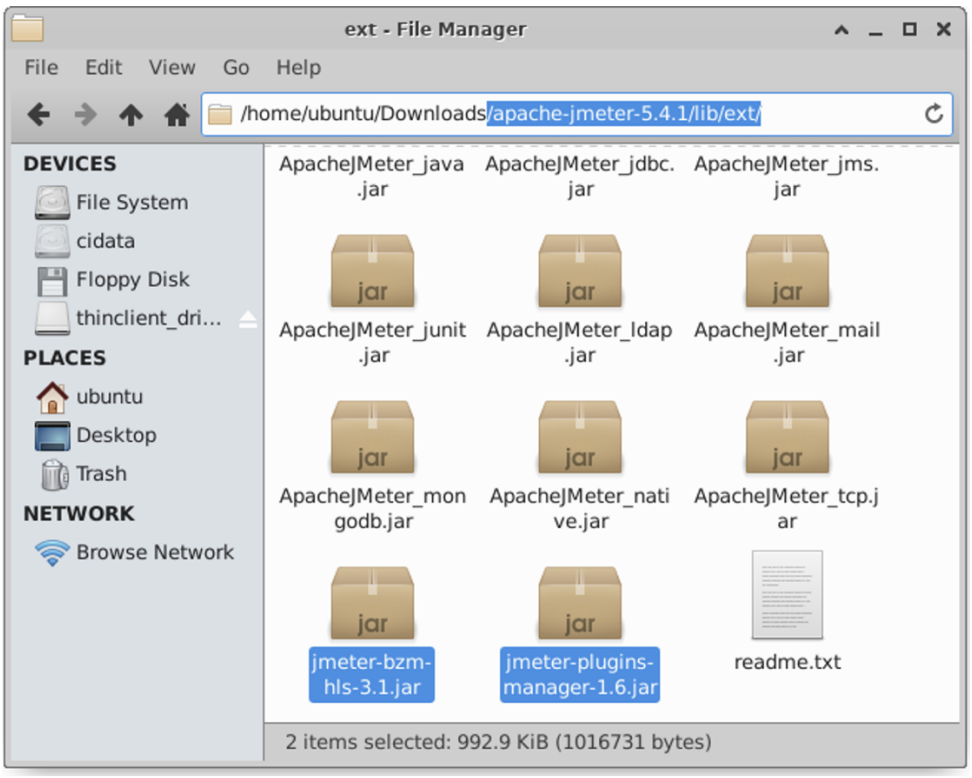


Close jMeter
Place jmeter-bzm-hls-3.1.jar into apache-jmeterFolder/lib/ext
Launch jMeter and see if it's Installed.
If you delete jar file from/lib/ext and restart jMeter, the plugin disappears. I tried doing this with jMeter Mongo DB jar file.



Last edited 25 March 2022.

End-State:



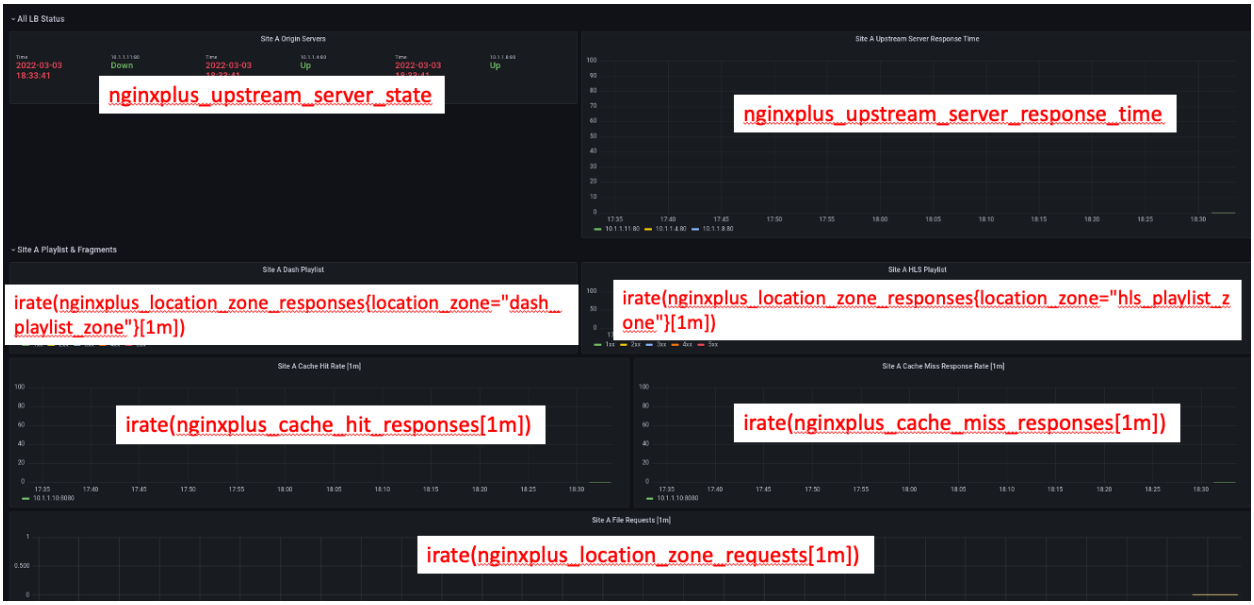
Last edited 25 March 2022.

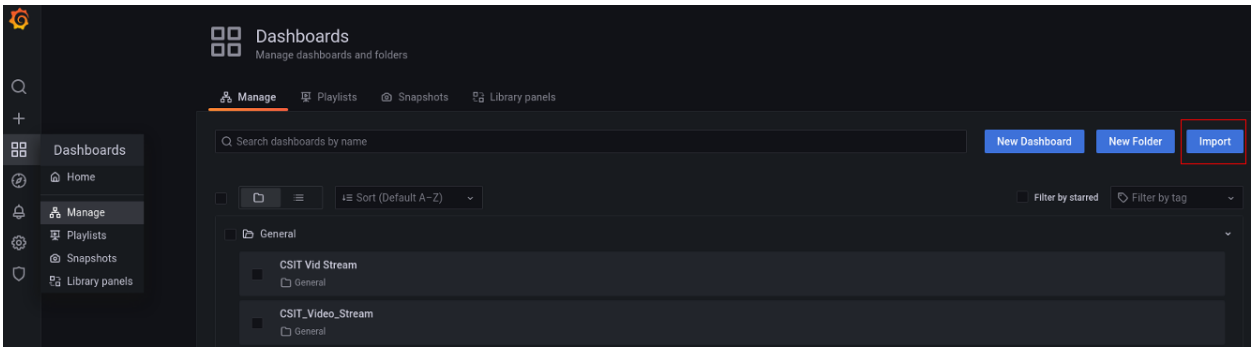
Grafana Dashboard (Optional)

As goodwill, we tried to help customer resolve their non-visibility of many sites. Therefore we showed a possible dashboard with Prometheus/Grafana.

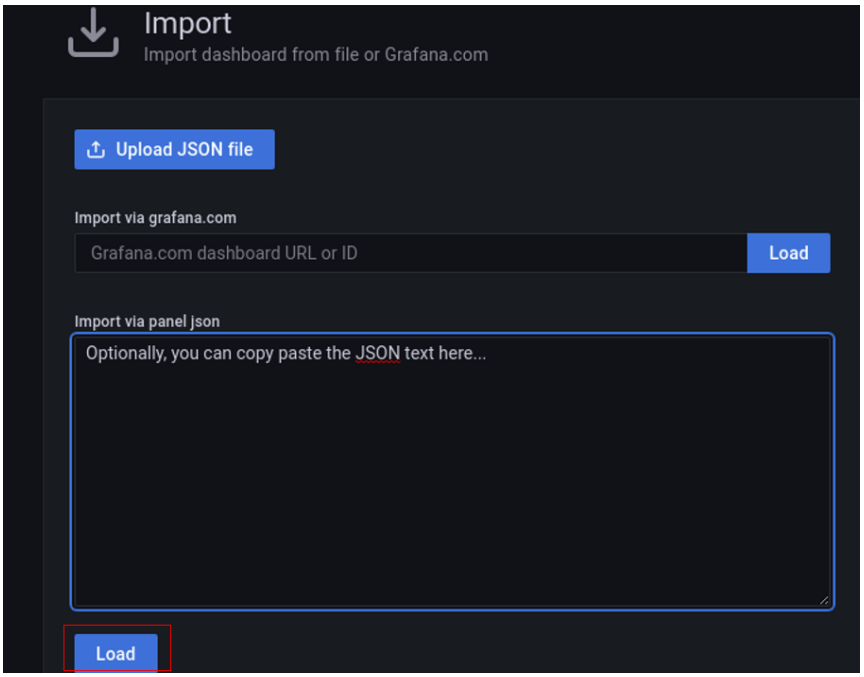
Install guide: https://www.nginx.com/blog/how-to-visualize-nginx-plus-with-prometheus-and-grafana/

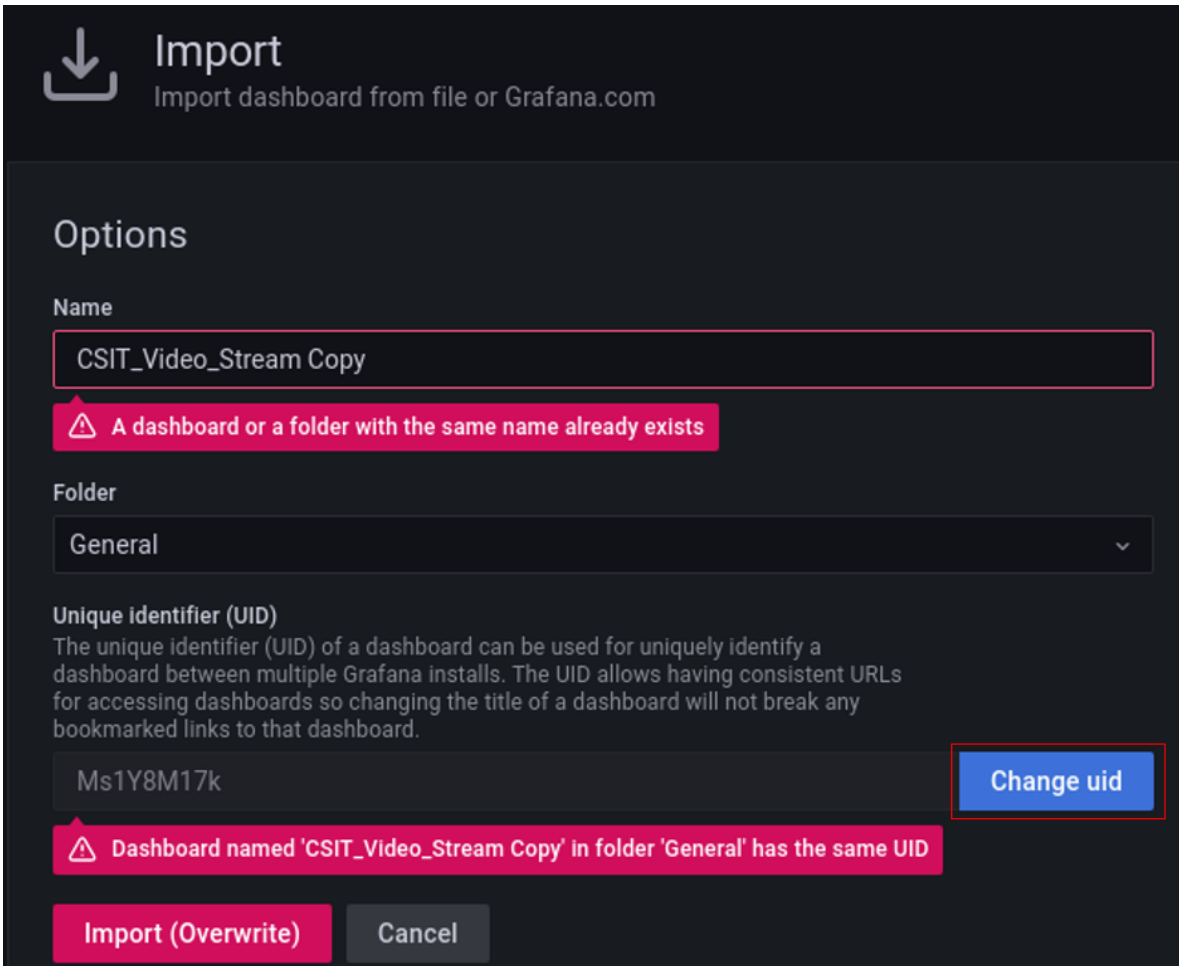
There is a JSON you can import Grafana dashboards and it was how we passed the dashboard for their own reference.





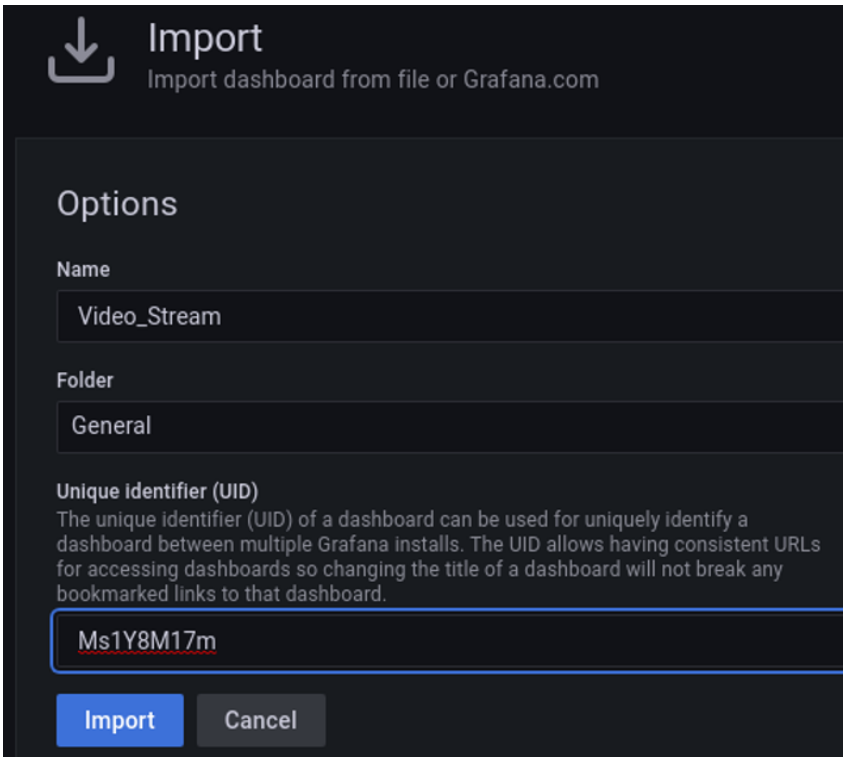
Last edited 25 March 2022.

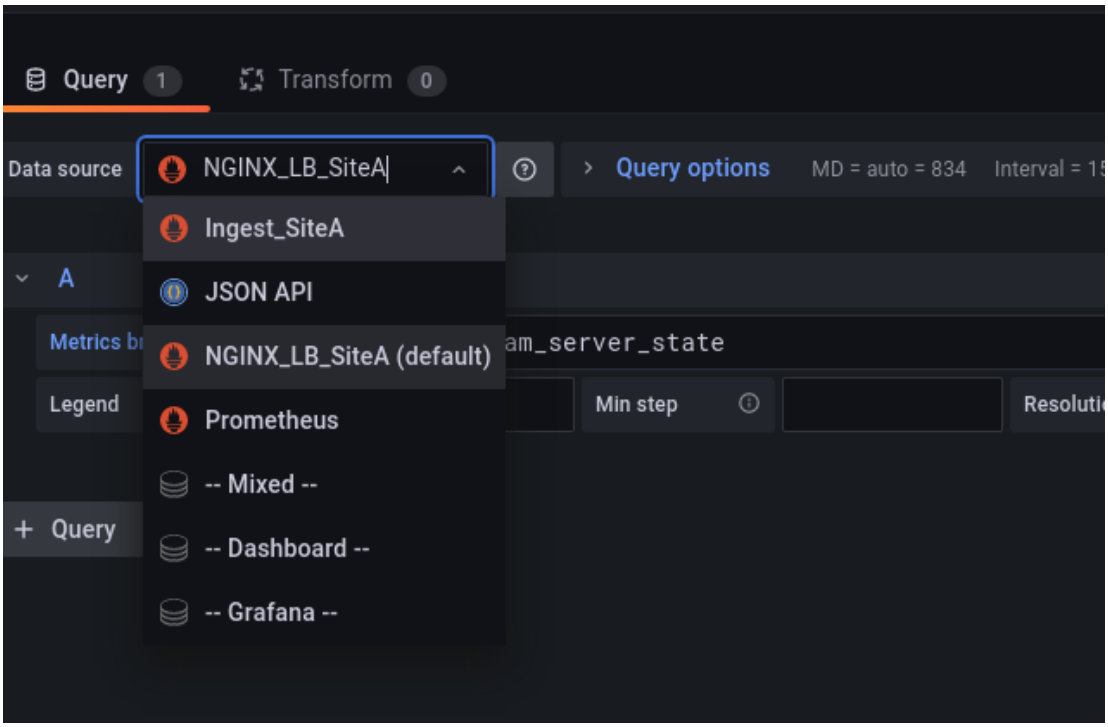




Rename the Title and UID. I randomly changed the last ‘k’ to ‘l’ or ‘m’, etc. Random I suppose.

Last edited 25 March 2022.





I suspect you need to change datasource. But you can see the query string for each dashboard element.