Intel® NFV Use Case: Content Delivery Network Release 1.0 User Guide

User Guide

Document Revision 1.0
Feb 2019

Disclaimers

Notice: this document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The benchmark results may need to be revised as additional testing is conducted. The results depend on the specific platform configurations and workloads utilized in the testing, and may not be applicable to any particular user's components, computer system or workloads. The results are not necessarily representative of other benchmarks and other benchmark results may show greater or lesser impact from mitigations.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548- 4725 or visit www.intel.com/design/literature.htm. No computer system can be absolutely secure.

Intel, SpeedStep, Intel Turbo Boost Technology, Xeon, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Figures

Tables

1.0 Introduction

Intel® Network Function Virtualization (NFV) Use Case package is a deployment infrastructure that helps users/ customers set up a NFV use case quickly out of box, with all necessary components in place and configured for optimal performance on an Intel hardware or platform. This release 1.0 is for NFV use case Content Delivery Network-CDN. The deployment infrastructure or CDN Play books released in this package pools the below components from their respective repositories, deploy it on the target machines, builds them, applies the configuration for better performance.

1. SVT : Scalable Video Technology, HEVC encoder.
2. ffmpeg : Video encode, decode, transcode libraries.
3. NGINX: Web Server
4. Apache Traffic Server: Web Server

This package also includes the performance configuration and additional configurations for a complete functional validation.

2.0 Reference Architecture

The NFV CDN package caters to three main sub use cases in a Visual Cloud in a box environment.

1. Media Transcode
2. Content Distribution-Streaming
3. Content Distribution-IPCDN

The below architecture diagram specifies a list of coworking but not necessarily inter dependent opensource software's that form a recipe for CDN use cases. The configurations specified in this user guide are for validation of various combination of functionalities and for optimal performance.

Intel's NFV infrastructure(NFVi) Hardware and software forms the base platform for the CDN stack.

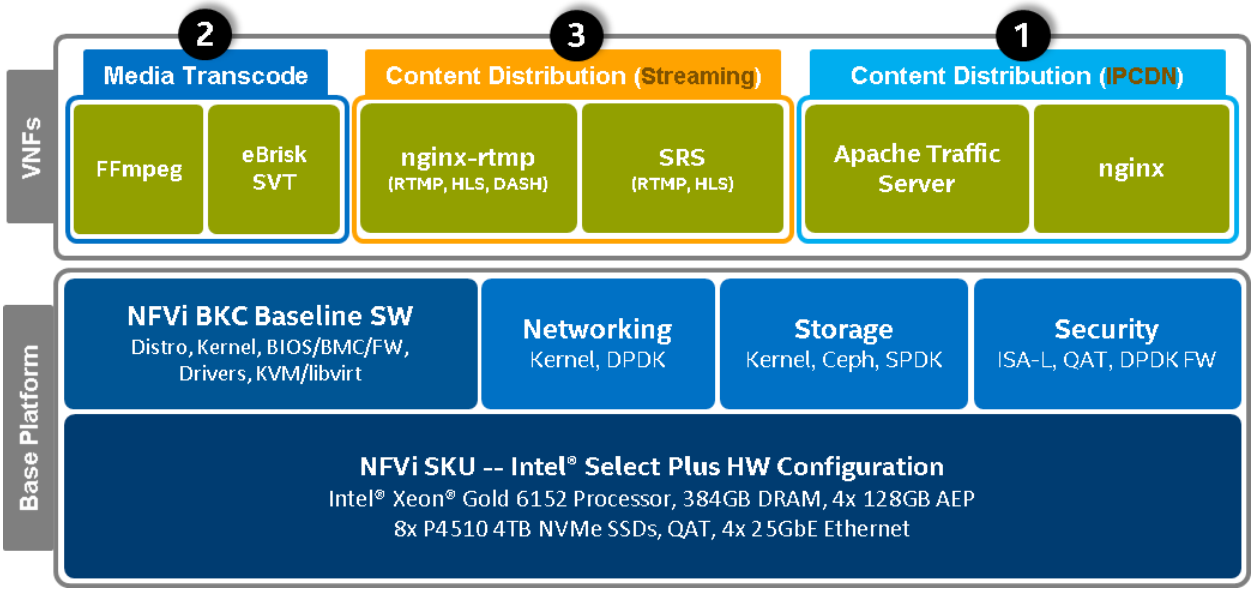


Figure 2-1 Intel® CDN Stack Diagram

***orchestrator or the control plane components are not part of this release.**

2.1 Intel® CDN Release 1.0 Distribution

Intel® NFV Use Case: CDN Release 1.0 is delivered in the form of Ansible playbooks on https: //01.org/. Intel® NFV Use Case: Content Delivery Network Release 1.0 Release Notes provides instructions on how to build and configure CDN components on Intel platforms.

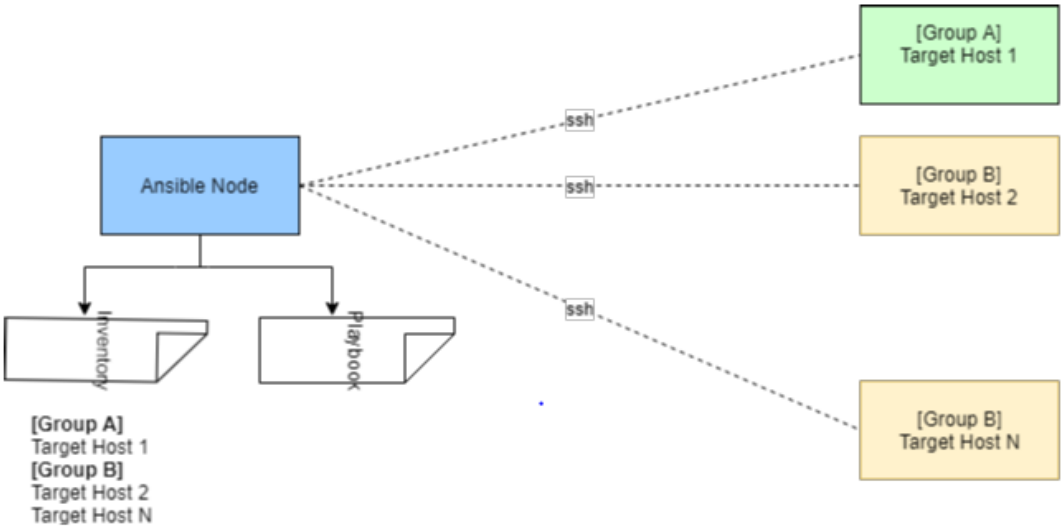


Figure 1-1 Overview of Ansible Configuration

3.0 Pre-requisites

The system under test is either a Intel Select Fast track kit or any other Intel Xeon HW (with HW configuration mentioned in the HW section)

- The server should have a working firmware to boot from USB. (https://networkbuilders.intel.com/intelselectfasttrackkit)
- A 8 GB (a maximum of 32 GB) USB pen drive
- Format the thumb drive to FAT32 format
- Prepare a Linux-installed system(Ubuntu 16.04)
- Prepare the software “Syslinux” (version 4.0 or even higher, version 6.3 is recommended)

4.0 System Configuration

The CDN package is built on the base NFV infrastructure published on Network Builder and the user can get the Intel select solutions fast track kit from https://networkbuilders.intel.com/intelselectfasttrackkit for better performance.

The user guide for the ISS kit can be found in
https://builders.intel.com/docs/networkbuilders/intel-select-fast-track-kit-for-nfvi-with-ubuntu-user-guide.pdf

4.1 HW configuration

One of the example system configuration assumed for all workloads described in this document are listed in table below. This configuration is similar to the INTEL® SELECT SOLUTION FOR NFVI PLUS CONFIGURATION HARDWARE

Component	Description
Processor	Skylake, Intel® Xeon® Gold 6152 Processor 30.25M Cache, 2.10 GHz OR Platinum 8180
Memory configuration	24x 16GB DDR4 2666MHz memory (384GB total), 4x P4500 1.2TB NVMe SSD's, 2x S4500 240GB SSD's (boot)
Network	4 port switch adapters, 4x dual 25G networking adapters, Intel® Ethernet Network Connection OCP I357-T4 (mgmt.)
Storage	150G/1.2T SSD

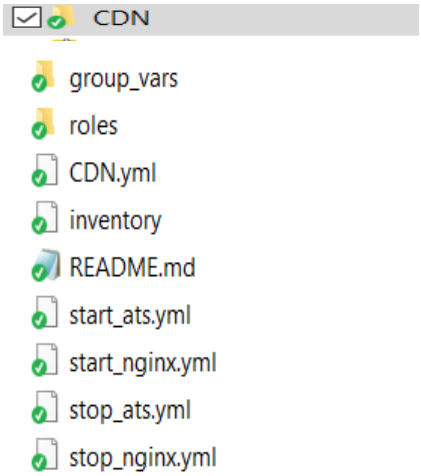
4.2 Software Configuration

Above the hardware level, there are many choices for firmware, operating systems, drivers, hypervisors, and other software components, which must be validated together to guarantee proper operation. The Intel® Select Fast Track Kit for NFVI, an instantiation of the Intel Select Solution for NFVI, uses Ubuntu Linux* as an option for its host operating system. An example instantiation of the reference design software stack is described below. Intel drives this firmware and software through a continuous validation process to ensure that software updates will perform correctly when integrated into a deployed system. The components of the software stack are subject to change. Apart from the NFVi software this package Pulls the below versions of opensource components.

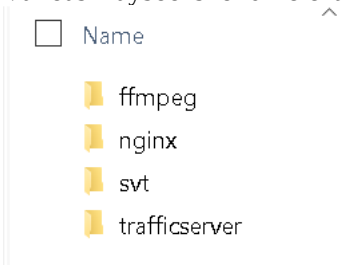
Component	Description
OS	Ubuntu18.04.1
Kernel	Ubuntu18.04.1-Kernel-4.15.0-38-generic
Libvirt	4.0.0
SPDK	18.10
SST	3.0.1054
collectd	5.8.0-master-2e83c33
Qemu	2.11.1
DPDK	18.08

i40evf	3.0.1-k
ixgbe	5.1.0-k
ATS-Traffic server	7.1.5
ffmpeg	4.0
SVT	V1.2.0

The ansible package delivered through this release has the below folder structure. It has playbooks for each individual component it will install, build and deploy.



Various Playbooks for different components,



- These playbooks deploy a implementation of various components of CDN such as ATS (version 7.1.5), ffmpeg (master), nginx (version 1.14.0)+ rtmp-module and SVT (master).
- To use CDN components, first edit group_vars/all (hostname of nginx node for creating ssl certificate and proxy environment).
- Also edit the "inventory" file that contains the hostnames/IPs and login credential of the machines on which you want these components to install.
- Ansible has host key checking enabled by default. If a host is reinstalled and has a different key in 'known_hosts', this will result in an error message until corrected. You can disable this behavior, you can do so by editing /etc/ansible/ansible.cfg

  [defaults]
  host_key_checking = False

- Alternatively this can be set by the ANSIBLE_HOST_KEY_CHECKING environment variable:

  $ export ANSIBLE_HOST_KEY_CHECKING=False

- Generate ssh key in the ansible machine, which we have to copy to all the remote hosts for doing deployments or configurations on them:

  $ ssh-keygen -t rsa -b 4096 -C "username@ip_add_of_ansible_machine"

  Now copy the ssh key generated to the remote hosts (Before copying the ssh key make sure that you are able to ssh the remote host where you want to copy the key):

  $ ssh-copy-id remote_user@remote_ip

  To set up SSH agent to avoid retyping passwords, you can do:

  $ ssh-agent bash
  $ ssh-add ~/.ssh/id_rsa

- For more detail about ansible setup refer the below link:

  https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html

- Now as we have setup our "inventory" file and other configurations are done, let's try pinging all the servers listed in the "inventory" file.

  $ ansible -i inventory all -m ping

  If pinging get successfull, then we are good and can go further :)

- For running the playbook go inside CDN directory where you have CDN.yml, inventory etc should be present. The ansible playbook command is given below:

  $ ansible-playbook -i inventory CDN.yml --become -K

  here -K, ask for privilege escalation password

- When the playbook run completes, the CDN components will be installed successfully on the target machines.

4.3 Installing Ansible using shell script

Ansible is open source software that automates software provisioning, configuration management, and application deployment.

The ansible_install.sh script provided will install the ansible and configure the target machines.

First make the script executable and then run in same directory like:

$ chmod +x ansible_install.sh

$ ./ansible_install.sh

- Script will install the dependencies required for ansible, and ansible version 2.7.1.
- It will also generate the ssh key in ansible machine (where ansible is installing) for this it will prompt for IP address username of ansible machine.

- Now it will prompt for how many target machines you want to configure.
- Then you have to enter the username and IP address of the target machines one by one, so that ssh key will successfully copied to these target machines.

If you are using ansible_install.sh script for installing ansible then you don't need to generate and copy ssh key manually.

4.4 Roles

Apache traffic server (ATS), FFMPEG, NGINX and SVT are the CDN workloads that will be installed by the Ansible plybook CDN.yml, CDN.yml includes all these components (ats, ffmpeg, nginx, svt) which are present in roles directory.

4.4.1 ATS

Apache Traffic Server is a high-performance web proxy cache that improves network efficiency and performance by caching frequently-accessed information at the edge of the network. This role install and configure the apache traffic server from source. After CDN.yml run completes, you can start trafficserver using start_ats.yml, before starting trafficserver you need to edit the remap.config in target machine at /opt/ats/etc/trafficserver/remap.config, edit the IP and port of your server, you can also edit storage.config, records.config and other config files accordingly which are present at /opt/ats/etc/trafficserver/*.
This Playbook will copy the optimal config files to target machines, if you want to edit any of config file you can do in target machine at /opt/ats/etc/trafficserver/* .

$ ansible-playbook -i inventory *start_ats.yml* --become -K

If you have made any changes in config files and want to restart the trafficserver then run the *stop_ats.yml*, like:
ansible-playbook -i inventory stop_ats.yml --become –K

Then start the trafficserver using *start_ats.yml*

vars directory contains the variable that are used in tasks/main.yml
handlers directory contains the handlers that are triggers by notify in tasks/main.yml
template directory contains the config files that will be copied to target machines

4.4.2 NGINX

NGINX is server for web serving, media streaming. In addition to its HTTP and HTTPS server capabilities. nginx+rtmp-module => Media streaming, http and https. This role install and configure the nginx+rtmp-module server from source.
This Playbook will copy the optimal nginx.conf files to target machines, if you want to make changes in conf file you can do that at /usr/local/nginx/conf/nginx.conf.

After CDN.yml run completes, you can start the nginx using *start_nginx.yml* inside nginx role, like:

$ ansible-playbook -i inventory *start_nginx.yml* --become -K

If you made any changes in conf files and want to restart the nginx then run the stop_nginx.yml, like:

$ ansible-playbook -i inventory *stop_nginx.yml* --become -K

Then start the nginx using *start_ats.yml* like above.
"vars" directory contains the variable that are used in tasks/main.yml
"handlers" directory contains the handlers that are triggers by notify in tasks/main.yml
"template" directory contains the conf file that will be copied to target machines

4.4.3 FFmpeg

FFmpeg is a command line tool for video and audio transcoding for both live and static content.

This role install and configure the ffmpeg from source.

ffmpeg is a very fast video and audio converter that can also grab from a live audio/video source.

It can also convert between arbitrary sample rates and resize video on the fly with a high

quality polyphase filter.

This playbook configured ffmpeg with following codecs:

- H.264 (libx264) video encoder
- H.265 (libx265) video encoder
- AV1 (libaom) video encoder/decoder
- VP8/VP9 (libvpx) video encoder/decoder

- AAC (libfdk-aac) audio encoder
- MP3 (libmp3lame) audio encoder
- Opus (libopus) audio decoder and encoder

For running the ffmpeg, go to localtion where input files are stored and run the ffmpeg command with or without codec:
$ ffmpeg -i input_files codec output_files

For more detail, please refer manual page of ffmpeg:
$ man ffmpeg

4.4.4 SVT

The Scalable Video Technology for HEVC Encoder (SVT-HEVC Encoder) is an HEVC-compliant encoder library core that achieves excellent density-quality tradeoffs, and is highly optimized for Intel Xeon Scalable Processor and on D processors. This role install ans build the SVT-HEVC from source.

1. For running SVT copy the binary (HevcEncoderApp, libHevcEncoder.so) from /opt/SVT-HEVC/Bin/Release to any location of your choice.

2. Change the permissions on the sample application "HevcEncoderApp" executable by running the command:

$ chmod +x HevcEncoderApp

3. cd into your chosen location

4. Run the sample application to encode.

5. Sample application supports reading from pipe.Eg.

$ ffmpeg -i [input.mp4] -nostdin -f rawvideo -pix_fmt yuv420p - |
./HevcEncoderApp -i stdin -n [number_of_frames_to_encode] -w [width] -h [height]

5.0 CDN Software Installation

Intel® CDN components can be installed using scripts associated with this release. Download the Intel® CDN Release 1.0 Scripts from 01.org. The scripts bundle contains the files shown in the table below:

Files	Description	Notes
CDN.yml	This playbook includes all the roles such as ats, nginx, ffmpeg and SVT. It will configure and install all the CDN components present in the roles directory.	
inventory	This file will contains the group names which will refer to the IPs of the target machines listed under particular group name.	Edit IPs of target machines
README.md	README.md file provides instructions for how to run the ansible script, what changes user has to made before running the scripts etc.	
group_vars/all	It contains the global variables like proxy environment variables, hostname of nginx server for SSL certificate.	Edit the proxy environment
roles/ffmpeg	This role will clone, configure and install the FFmpeg (n4.0.1) from source and also install the dependencies required for FFmpeg.	
roles/nginx	This role will clone, configure and install the nginx (version 1.14.0) with rtmp module from source. It will also install the dependencies required for nginx.	
roles/svt	This role will install dependencies, clone the SVT (v1.2.0) form github and build it.	
roles/trafficserver	This role will install the dependencies, clone the ATS(7.1.5) from github, configure and install the trafficserver.	
ansible_install.sh	This is a shell script for installing and configuring ansible, it also generate ssh-key. It will ask for number of remote machines and then copy the ssh-key to them.	
start_ats.yml	After successfully install the ATS using CDN.yml, you can start trafficserver using this playbook.	
stop_ats.yml	If you want to modify the config file or want to restart the trafficserver, first stop the trafficserver using this playbook and start again using start_ats.yml playbook.	
start_nginx.yml	This playbook will start the nginx server once you have successfully install the nginx using CDN.yml.	
stop_nginx.yml	You can stop the nginx server using this playbook. If you want to make any changes in config file or want to restart the nginx server then you need to stop and again start the nginx server after making changes in config file.	

6.0 Installation Steps for FFmpeg

FFmpeg is a command line tool for video and audio transcoding for both live and static content. FFmpeg is a very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high quality polyphase filter.

6.1 Build Dependencies

1. nasm.
2. GCC
3. git
4. libx264-dev
5. CMake
6. OS – Ubuntu

6.2 Installation

1. Clone the FFmpeg (master) from https://git.ffmpeg.org/ffmpeg.git

   $ git clone https://git.ffmpeg.org/ffmpeg.git
2. $ ./configure #Run configure file
3. $ sudo make #To build ffmpeg
4. $ sudo make install #To install all binaries and libraries

After successful installation of ffmpeg, we can check version using following command:

- *$ ffmpeg --version*
- *$ man ffmpeg* #For manual manual page of ffmpeg

6.3 Running FFmpeg

Below is the example for running FFmpeg:

- *Go to the directory where your input videos are located.*
- *Make sure all input files are copied before running below command.*
- *ffmpeg –i inputfile.mp4 outfile.mkv*

7.0 Installation Steps for SVT

7.1 Build Dependencies

- nasm
- YASM Assembler version 1.2.0 or later
- gcc 5.4.0 or later
- CMake 3.5.1 or later
- OS – Ubuntu

7.2 Build Instructions

- cd Build/linux
- ./build.sh
- Binaries can be found under Bin/Release and / or Bin/Debug

7.3 Installation

- For the binaries to operate properly on your system, the following conditions have to be met:

- Copy the binaries under a location of your choice.
- Change the permissions on the sample application "HevcEncoderApp" executable by running the command:

  $ chmod +x HevcEncoderApp

- cd into your chosen location
- Run the sample application to encode.
- Sample application supports reading from pipe. E.g.

  $ ffmpeg -i [input.mp4] -nostdin -f rawvideo -pix_fmt yuv420p - |
  ./HevcEncoderApp -i stdin -n [number_of_frames_to_encode] -w [width] -h [height].

8.0 Installation Steps for ATS

Apache Traffic Server is a Cache server. It's set up and run needs a 2-node setup as below:

Node1: ATS Caching Proxy + Content Server (collocated)
Node2: HTTP Traffic Generator

8.1 Build Dependencies

In order to build Traffic Server from source you will need the following development tools and libraries installed:

- autoconf
- automake
- libtool
- pkg-config
- libmodule-install-perl
- gcc/g++ or clang/clang++
- libssl-dev
- tcl-dev
- libpcre3-dev
- libcap-dev (optional, highly recommended)
- libhwloc-dev (optional, highly recommended)
- libncurses5-dev (optional, required for e.g.: traffic_top)
- libcurl4-openssl-dev (optional, required for e.g.: traffic_top)
- flex (optional, required for e.g. WCCP)

8.2 Installation

Preparing the Source Tree

```
autoreconf -if
```

Configuration Options

1. A configure script will be generated from configure.ac which may now be used to configure the source tree for your build.

```
./configure --prefix=/opt/ats
```

2. Once the source tree has been configured, you may proceed on to building with the generated Makefiles. The make check command may be used to perform sanity checks on the resulting build, prior to installation, and it is recommended that you use this.

```
make
make check
```

3. With the source built and checked, you may now install all of the binaries, header files, documentation, and other artifacts to their final locations on your system.

```
sudo make install
```

4. Finally, it is recommended that you run the regression test suite. Please note that the regression tests will only be successful with the default layout.

```
cd /opt/ats
sudo bin/traffic_server -R 1
```

8.3 Configuration Changes:

8.3.1 records.cofig

/opt/ats/etc/trafficserver/records.config
Edit the RAM cache and RAM cache cutoff size accordingly:

```
CONFIG proxy.config.cache.threads_per_disk INT 16
CONFIG proxy.config.cache.ram_cache.size INT 50G
CONFIG proxy.config.cache.ram_cache_cutoff INT 5G
```

CONFIG proxy.config.cache.limits.http.max_alts INT 5
CONFIG proxy.config.body_factory.template_sets_dir STRING etc/trafficserver/body_factory
CONFIG proxy.config.url_remap.filename STRING remap.config
CONFIG proxy.config.ssl.server.ticket_key.filename STRING NULL
CONFIG proxy.config.http.cache.required_headers INT 1
CONFIG proxy.config.url_remap.pristine_host_hdr INT 1
CONFIG proxy.config.cache.ram_cache.use_seen_filter INT 1
CONFIG proxy.config.cache.ram_cache.algorithm INT 0
CONFIG proxy.config.cache.ram_cache.compress INT 1

8.3.2 remap.config

/opt/ats/etc/trafficserver/remap.config

map client_url origin_server_url
map / origin_server_url #define a catchall for requests

8.3.3 storage.config

/opt/ats/etc/trafficserver/storage.config
Edit according to your system if it is

var/trafficserver 50G #change accordingly

you can edit your disk path and size, it's not mandatory
/media/ramdisk 100G volume=1 id=ramdisk
/media/ramdisk1 200G volume=1 id=ramdisk1
/home/ubuntu 90G volume=2 id=mount

If you are using volume numbers in storage.config then you can edit below config files:

8.3.4 volume.config

# /opt/ats/etc/trafficserver/volume.config
volume=1 scheme=http size=80%
volume=2 scheme=http size=20%

8.3.5 hosting.config

# /opt/ats/etc/trafficserver/hosting.config
hostname=http://localhost volume=1,2
hostname=server volume=x

ATS can get started using below command:

$ /opt/ats/bin/trafficserver start

Debug logs under /var/log/trafficserver

For more information kindly refer the https://docs.trafficserver.apache.org/en/latest/index.html .

9.0 Installation Steps for NGINX

NGINX is server for web serving, media streaming. In addition to its HTTP and HTTPS server capabilities.

Node1: NGINX+rtmp module=> Media streaming, http and https.
NGINX server can be installed on bare metal or VM.

9.1 Build Dependencies

In order to build nginx server with rtmp-module on Ubuntu from source you will need the following development tools and libraries installed:

7. build-essential
8. libpcre3
9. libpcre3-dev
10. libssl-dev

Clone nginx:

git clone https://github.com/intel/SVT-HEVC.git

Clone nginx-rtmp-module:

git clone https://github.com/sergey-dryabzhinsky/nginx-rtmp-module.git

9.2 Configuration and Installation

11. cd nginx
12. /auto/configure --with-http_ssl_module --add-module=../nginx-rtmp-module
13. make
14. make install

By default nginx installed in /usr/local/nginx/ and we can test if it installed correctly or not by running this command:

$ /usr/local/nginx/sbin/nginx

If it executed without errors, we can see the results by adding our localhost IP 127.0.0.1 or the name localhost/IP in our browser.

Also, we can edit the conf file and add at the bottom of the nginx's config file the rtmp configuration that we want to use on our host (for our streaming).

$ vi /usr/local/nginx/conf/nginx.conf

9.3 Create nginx configuration file

1. **rtmp module config:**

```
rtmp {
    server {
            listen 1935; # Listen on standard RTMP port
            chunk_size 4000;

            application live {
            live on;
            # Turn on HLS
            hls on;
            hls_path /mnt/hls/;
            hls_fragment 3;
            hls_playlist_length 60;
            # disable consuming the stream from nginx as rtmp
            deny play all;
        }
    }
}
```

2. **http/https server config:**

Before configure https server need to be create certificate and key file.
Create the certificate:
Change to the root user and change to the directory in which you want to create the certificate and key pair. That location will vary depending on your needs.

15. sudo su
16. mkdir /root/certs && cd /root/certs
17. openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out MyCertificate.crt -keyout MyKey.key

    You will be prompted to add identifying information about your website or organization to the certificate. Since a self-signed certificate won't be used publicly.

18. cp MyCertificate.crt MyKey.key /usr/local/nginx.conf

Now next step is http/https configuration:

```
    http {
        include       mime.types;
          tcp_nopush on;
          directio 512;
          default_type  application/octet-stream;

          sendfile        off;
          keepalive_timeout  65;

          server {
             listen       80;
             listen      443 ssl;
             server_name  localhost;

             ssl_certificate      MyCertificate.crt;
             ssl_certificate_key  MyKey.key;

             ssl_session_cache    shared:SSL:1m;
             ssl_session_timeout  5m;

             ssl_ciphers  HIGH:!aNULL:!MD5;
             ssl_prefer_server_ciphers  on;

   location /hls {
             # Disable cache
             add_header Cache-Control no-cache;

             # CORS setup
             add_header 'Access-Control-Allow-Origin' '*' always;
             add_header 'Access-Control-Expose-Headers' 'Content-Length';

             if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain charset=UTF-8';
                add_header 'Content-Length' 0;
                return 204;
             }

             types {
                application/vnd.apple.mpegurl m3u8;
                video/mp2t ts;
             }

             root /mnt/;
          }
        }
```

10.0 Notes on Performance Configurations and Work Load Behavior

SVT- Scalable Video Technology –HEVC Encode

- Workload performs nearly same in a virtual machine as compared to bare metal.
- FPS and Latency numbers for both VM and Bare metal are nearly same.
- Workload performs better when pinned to individual sockets.
- Workload takes benefit of platform turbo and Hyperthreading.
- Workload is more core bound.

ffmpeg

- WL performance is very same for Bare Metal vs VM
- Need to use config –vcode –libx264 to get the better performance in VM (do not take it by default)
- Workload takes benefit of platform turbo and Hyperthreading.
- Workload is more core bound.

VM Configuration

VM	Single FAT VM	Multiple thin VMs
RAM (GB)	100	20-50
VCPU	88	10-30
DISK SIZE(GB)	105	20-50

NGINX

Refer to the section 9 for performance configuration.

ATS

Performance configuration is already getting applied in the automation. Please refer section 8.

10.0 Additional Configuration for Complete Functional Validation

The Ansible playbook takes care of the performance configuration for each of the components. If someone wants to do a complete validation of the different components, they can follow the below additional configuration used for functional validation.

10.1 ATS

Build Dependencies

In order to build Traffic Server from source you will need the following development tools and libraries installed:
You can use the following command line to install most of the dependencies

apt-get install -y pkg-config libtool gcc g++ make openssl libssl-dev hwloc lua5.3 libpcre3 libpcre3-dev flex
libreadline-dev libncurses5-dev tcl-dev tcl curl clang-tidy autoconf
1) The following steps must be execute manually:
2) libcap
a. download latest version from http://www.tcpdump.org/#latest-releases
b. tar zxvf libpcap-*.*.*.tar.gz
c. cd zxvf libpcap-*.*.*
d. #./configure & make & make install
e. Configure: error: Neither flex nor lex was found. - apt-get install flex
f. configure: error: yacc is insufficient to compile libpcap - apt-get install bison
3) update gcc&g++
a. sudo add-apt-repository ppa:ubuntu-toolchain-r/test
b. sudo apt-get update
c. sudo apt-get -y install gcc-7
d. sudo apt-get -y install g++-7
e. sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 100
f. sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-7 100
g. sudo update-alternatives --config gcc
h. sudo update-alternatives --config g++
i. if "bash: add-apt-repository: command not found" please apt-get install software-propertiescommon
4) update CURL to support http2
a. install dependencies
i. sudo apt-get -y install git g++ make binutils autoconf automake autotools-dev libtool pkg-config zlib1g-dev libcunit1-dev libssl-dev libxml2-dev libev-dev libevent-dev libjansson-dev libjemalloc-dev cython python3-dev python-setuptools
b. make and install nghttp2
i. git clone https://github.com/tatsuhiro-t/nghttp2.git
ii. cd nghttp2
iii. autoreconf -i
iv. automake
v. autoconf
vi. ./configure
vii. make
viii. sudo make install
c. update CURL version
i. wget https://curl.haxx.se/download/curl-*.*.*.tar.gz
ii. tar -xvjf curl-*.*.*.tar.bz2
iii. cd curl-*.*.*
iv. ./configure --with-nghttp2=/usr/local --with-ssl
v. sudo make && make install
vi. echo '/usr/local/lib' > /etc/ld.so.conf.d/local.conf
vii. ldconfig
d. using curl -V Check that CURL supports HTTP2

```
root@e5-lijing:/home/ats/trafficserver-8.0.0# curl -V
curl 7.61.1 (x86_64-pc-linux-gnu) libcurl/7.61.1 OpenSSL/1.0.2g zlib/1.2.8 nghttp2/1.35.0-DEV
Release-Date: 2018-09-05
Protocols: dict file ftp ftps gopher http https imap imaps pop3 pop3s rtsp smb smbs smtp smtps telnet tftp
Features: AsynchDNS IPv6 Largefile NTLM NTLM_WB SSL libz TLS-SRP HTTP2 UnixSockets HTTPS-proxy
```

e. If not show HTTP2 please log in again.

Once ATS installation is done,
1) Preparing the Source Tree
2) autoreconf -if
3) Configuration Options
4) ./configure --prefix=/opt/ats --with-user=pid --with-group=pid
a. NOTE: --with-user= --with-group= please input the name that exists else when make install will show:
b. if [`id -un` != "root"]; then \
i. /usr/bin/install -c -d /opt/ats/var /opt/ats/var/log/trafficserver /opt/ats/var/trafficserver \
ii. /etc/trafficserver /opt/ats/share/trafficserver /opt/ats/var/trafficserver; \
c. else \
i. /usr/bin/install -c -d /opt/ats/var; \
ii. /usr/bin/install -c -d -o trafficserver -g trafficserver /opt/ats/var/log/trafficserver \
iii. /opt/ats/var/trafficserver /etc/trafficserver \
iv. /opt/ats/share/trafficserver /opt/ats/var/trafficserver; \
d. fi
e. /usr/bin/install: invalid user ‘trafficserver’
f. Makefile:1133: recipe for target 'install-data-local' failed
8) sudo make && make install
9) Finally, it is recommended that you run the regression test suite. Please note that the regression tests will only be successful with the default layout.
a. cd /opt/ats/bin
b. sudo ./traffic_server -R 1
c. If shows “REGRESSION_TESTDONE: PASSED” was installed successfully
10) Run ATS:
a. # Usage /opt/ats//bin/trafficserver {start|stop|status|restart}

3. Run ATS test

Before running the autest, please follow below steps:

- Enter in root mode
  $sudo su
- Set the environment proxy
- Make sure trafficserver is stop if not, to check the status
  $/opt/ats/bin/trafficserver status

  If it is running then stop using:
  $ /opt/ats/bin/trafficserver stop
- Run bootstrap.py from /opt/trafficserver/tests/
  #./bootstrap.py
- Come out from root mode
  #exit
- Clear the environment proxy and trafficserver server should be stopped if it is running
  $unset http_proxy
  $unset https_proxy
  $ /opt/ats/bin/trafficserver stop
- If you don’t want to display debug information on the screen type:
  $set +x
  You can undo it using:
  $set -x

- Now you can run the autest.sh
  $./autest.sh -D /opt/trafficserver/tests/gold_tests --sandbox /tmp/sb --ats-bin /opt/ats/bin

Build Dependencies

NGINX is server for web serving, media streaming. In addition to its HTTP and HTTPS server capabilities.

Node1: NGINX+rtmp module=> Media streaming, http and https.

NGINX server can be installed on bare metal.

Hardware - Wolf pass platform with OS-Ubuntu installed.

Software -- Go to NFV_PQ_PKG_0.1.0-74\Web\nginx

In order to build nginx server with rtmp-module on Ubuntu from source you will need the following development tools and libraries installed:

Clone nginx-rtmp-module: git clone https://github.com/sergey-dryabzhinsky/nginxrtmp-module.git

Install below nginx dependencies

- **build-essential**
- **libpcre3**
- **libpcre3-dev**
- **libssl-dev**

2. NGINX Install and Configuration

Move the configure file out of auto directory and give the full permission.

root@unassigned:/home/pid/nginx/auto# mv configure /home/pid/nginx/

root@unassigned:/home/pid/nginx# chmod 777 configure

1). Compile nginx from the configure script file

root@unassigned:/home/pid/nginx# ./configure --add-module=../nginx-rtmp-module-master --withhttp_v2_module --with-http_ssl_module --with-http_realip_module --with-http_sub_module --withhttp_slice_module --with-http_dav_module --with-debug --with-http_auth_request_module --withhttp_addition_module --with-http_gzip_static_module --with-http_gunzip_module --with-stream --withmail --with-http_flv_module --with-http_stub_status_module --with-mail_ssl_module --withstream_ssl_module --with-stream_realip_module --with-http_secure_link_module --withstream_ssl_preread_module --with-http_perl_module --http-uwsgi-temp-path=/usr/local/nginx/uwsgi --
http-scgi-temp-path=/usr/local/nginx/scgi --with-http_image_filter_module --with-stream_geoip_module --with-http_geoip_module --with-http_image_filter_module --with-stream_geoip_module --withhttp_geoip_module --with-http_xslt_module

Notice the --add-module=../nginx-rtmp-module argument, the path must point correctly to the cloned module

Notice: above module installation are needed when doing nginx test

2). Once the source got configured, you may proceed on to building with the generated make.

make –j 1

• (Optional) replace -j 1 with the amount of cpu's on your computer to accelerate the compilation

Now install all of the binaries, header files, documentation, and other artifacts to their final locations on your system.

sudo make install

By default in the compiled nginx installed in /usr/local/nginx/ and we can test if it installed correctly by running this
command:

sudo /usr/local/nginx/sbin/nginx

If it executed without errors, we can see our results by adding our localhost IP 127.0.0.1 or the name localhost/IP in
our browser.
Also, we can edit the conf file and add at the bottom of the nginx's config file the rtmp configuration that we want to
use on our host (for our streaming).
sudo vi /usr/local/nginx/conf/nginx.conf

3). Create nginx configuration file:

i. rtmp module config:

```
worker_processes auto;
events {
worker_connections 1024;
}
rtmp {
server {
listen 1935; # Listen on standard RTMP port
chunk_size 4000;
application live {
live on;
# Turn on HLS
hls on;
hls_path /mnt/hls/;
hls_fragment 3;
hls_playlist_length 60;
# disable consuming the stream from nginx as rtmp
deny play all;
}
}
}
```

Note that the example point /mnt/hls/ as the target path for the hls playlist and video files.
You can change this to a different directory but make sure that nginx have write permissions.

ii. http/https server config:

Before configure https server need to be create certificate and key file.
Create the certificate:
Change to the root user and change to the directory in which you want to create the certificate and key pair.
That location will vary depending on your needs.

```
su –root
mkdir /root/certs && cd /root/certs
```

Then,

```
root@unassigned:~/certs# openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out
MyCertificate.crt -keyout MyKey.key
```

You will be prompted to add identifying information about your website or organization to the certificate.
Since a
self-signed certificate won’t be used publicly,

Copy the certificate and key files to below path:

root@unassigned:~/certs# cp MyCertificate.crt MyKey.key /usr/local/nginx/conf/

Now next step is http/https configuration:-

```
http {
include mime.types;
directio 512;
default_type application/octet-stream;
sendfile on;
tcp_nopush on;
keepalive_timeout 65;
server {
listen 80;
listen 443 ssl;
server_name localhost;
ssl_certificate MyCertificate.crt;
ssl_certificate_key MyKey.key;
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 5m;
ssl_ciphers HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers on;
location /hls {
# Disable cache
add_header 'Cache-Control' 'no-cache';
# CORS setup
add_header 'Access-Control-Allow-Origin' '*' always;
add_header 'Access-Control-Expose-Headers' 'Content-Length';
if ($request_method = 'OPTIONS') {
add_header 'Access-Control-Allow-Origin' '*';
add_header 'Access-Control-Max-Age' 1728000;
add_header 'Content-Type' 'text/plain charset=UTF-8';
add_header 'Content-Length' 0;
return 204;
}
types {
application/dash+xml mpd;
application/vnd.apple.mpegurl m3u8;
video/mp2t ts;
}
root /mnt/;
}
}
}
```

FFMPEG - TRANSCODING

FFmpeg is a command line tool for video and audio transcoding for both live and static content.

1. Build Dependencies:

• nasm and yasm .
• OS – Ubuntu

2. Installation & Configuration:

1. Unzip NFV_PQ_PKG_0.1.0-68.zip.
2. Go to media/ffmpeg folder
3. $ ./configure(run configure file).
4. $sudo make (To build ffmpeg).
5. $sudo make install(to install all binaries and libraries).
6. ./configure --enable-libx264 --enable-libx265 --enable-gpl --enable-libwebp --disable-optimizations --samples=../fate-suite --enable-openssl --enable-nonfree --enable-libzmq --enable-libkvazaar --enablelibvpx --enable-libvorbis --enable-libsrt --enable-libmfx

SVT -ENCODER

1. Make sure to install the dependencies (cmake and yasm)
2. give full permission to build.sh (chmod 777 build.sh)
3. run ./build.sh now
4. The sources are already built and binaries are already present under Bin/Release and /or Bin/Debug.
5. The Build folder contain two sub-folders "linux" and "content"
6. Content folder contains the video files and Linux folder contains the test scripts(SpeedTest.sh) for running the
workload.
7. In folder “linux” HevcEncoderApp file and libHevcEncoder.so file should be present to start the application.(copy these two files from Bin/Release or Bin/Debug)
8. Give the full permission for the file HevcEncoderApp file (chmod 777 HevcEncoderApp)
9. The encoder parameters present are listed in this table below along with their status
of support, command line parameter and the range of values that the parameters can take.

Encoder Parameter as shown in the configuration file	**Command Line parameter**	**Range**	**Default**	**Description**
-nch	[1 - 6]	1	Number of encode instances	
-c	any string	null	Configuration file path	
InputFile	-i	any string	null	Input file path and name
StreamFile	-b	any string	null	output bitstream file p
ErrorFile	-errlog	any string	stderr	error log displaying configuration or encode errors
UseQpFile	-use-q-file	[0,1]	0	When set to 1, overwrite the picture qp assignment using qp values in QpFile
QpFile	-qp-file	any string	null	Path to qp file
EncoderMode	-encMode	[0 - 12]	9	A preset defining the quality vs density tradeoff point that the encoding is to be performed at. (e.g. 0 is the highest quality mode, 12 is the highest

				density mode). Section 3.4 outlines the preset availability per resolution
Tune	-tune	[0,1]	1	0=SQ - subjective quality mode, 1=OQ - objective quality mode
LatencyMode	-latency mode	[0,1]	0	For lower latency (0: Normal Latency, 1: Low Latency)
EncoderBitDepth	-bit-depth	[8 , 10]	8	specifies the bit depth of the input video
CompressedTenBitFormat	- compressed ten-bit format	[0,1]	0	Offline packing of the 2bits: requires two bits packed input (0: OFF, 1: ON)
SourceWidth	-w	[64 - 8192]	0	Input source width
SourceHeight	-h	[64 - 4320]	0	Input source height
FrameToBeEncoded	-n	[0 - 2^31 - 1]	0	Number of frames to be encoded, if number of frames is > number of frames in file, the encoder will loop to the beginning and continue the encode. 0 encodes the full clip.
BufferedInput	-nb	[-1, 1 to 2^31 -1]	-1	number of frames to preload to the RAM before the start of the encode If -nb = 100 and –n 1000 -- > the encoder will encode the first 100 frames of the video 10 times Use -1 to not preload any frames.
rofile	-profile	[1,2]	2	1: Main, 2: Main 10
Tier	-tier	[0,1]	0	0: Main, 1: High

NOTICES AND DISCLAIMERS

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INTEL DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, AS WELL AS ANY WARRANTY ARISING FROM COURSE OF PERFORMANCE, COURSE OF DEALING, OR USAGE IN TRADE.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at Intel.com.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit **www.intel.com/benchmarks**.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.
Notice Revision #20110804

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting **www.intel.com/design/literature.htm**.

Intel, the Intel logo and others are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others