IBM Aspera HSTS as tethered node in AoC

2025/09/30

Contents

Chapter 1

Introduction

The procedure is documented in the Aspera on Cloud manual:

https://ibmaspera.com/help/0_tethered_map

https://www.ibm.com/docs/en/aspera-on-cloud?topic=node-tether-your-aspera-transfer-server-aspera-cloud

The procedure below is similar.

Instead of a metered transfer server license, we use here a license file. This is adapted for evaluations or to use a perpetual license.

This procedure is especially adapted to set up a self-managed Aspera High-Speed Transfer Server (HSTS) or Aspera High-Speed Transfer Endpoint (HSTE) as a tethered node to Aspera on Cloud (AoC) for a Proof of Concept (PoC) or evaluation.

> **Note**
>
> For Aspera Endpoint (HSTE), transfers with Connect Client version 3 is not supported. Also, a special configuration is required (see later)

1.1 Assumptions

The VM where HSTS will run has a direct internet connection (no forward, no reverse proxy): it can reach internet, and can be reached from internet. If NAT is used for the Node API, then we assume here that the same port is used for external and internal, else both ports shall be listened by Nginx so that both external and internal users can reach it. If proxies are used/needed, then additional configuration can be done, not covered here.

> **Note**
>
> It is also possible to use HTTPS instead of SSH for the TCP connection for transfers. In that case, a single HTTPS port may be shared between node and transfer. That requires additional configuration in Nginx.

1.2 Pre-requisites

In order to tether a self-managed node to Aspera on Cloud, the following are requited:

- A self-managed Linux system with admin (`root`) access (e.g. Rocky 9)
- Official hardware requirements, typically 4 cores, 8GB RAM
- A public IP address
- The server is reachable on a minimum of 2 TCP ports (for Node : 443 and SSH : 33001) and 1 UDP port (for FASP : 33001), so typically TCP/443 TCP/33001 UDP/33001 (configurable)
- A DNS A record (FQDN) for that IP address (or use FreeDNS, see below)
- A TLS certificate for that FQDN (or use `letsencypt` see below: this requires port TCP/80)

- A license file provided by IBM. For example, an evaluation license file:

  `87650-AsperaEnterprise-unlim.eval.aspera-license`

- The installation package for HSTS: for example:

  `ibm-aspera-hsts-4.4.5.1646-linux-64-release.rpm`

> **Note**
>
> The installation is also possible on other OS (macOS, Windows, ...), but this manual focusses on Linux.

1.3 Download the HSTS RPM

To download the RPM for HSTS (or HSTE), one can use the following methods:

- If you are an IBMer or have access to the Aspera downloads:
  - Go to https://ibm.com/aspera
  - Navigate to Download and Documentation, and then Server
  - Select Download Now for HSTS

    That bring to Fix Central
  - Click on the desired HSTS or HSTE version, and then make sure to select HTTP Download
  - Then right-click on the RPM link, and do Copy link location

    This represents a temporary direct download URL
  - Then follow the instructions below
- If IBM provided with a private link to fix central:
  - Navigate to the provided private link
  - Click on the desired HSTS version, and then make sure to select HTTP Download
  - Then right-click on the RPM link, and do Copy link location

    This represents a temporary direct download URL
  - Then follow the instructions below
- If you were provided with the direct download link (temporary), just follow the instructions below

On Linux execute:

```
wget [URL link from previous step here]
```

Alternatively, if `wget` is not available, `curl` is always present:

```
curl -o [paste only the file name of RPM] [paste the full link here]
```

For the license file, you can directly `vi` on Linux, and paste inside. Alternatively, use `scp` to transfer those files.

You will set the path to those two files in the variables in next section.

1.4 DNS record

A FQDN (DNS A Record) is required for the public address of the HSTS.

If none is defined, it is possible to use a free service like FreeDNS for that (PoC).

Use a domain in freedns that has lower number of users, so that you are not restricted when you'll generate the `letsencrypt` cert.

Chapter 2

Installation and configuration of tethered node

We assume here that a compatible Virtual (or physical) Machine is installed with a RHEL-compatible Linux distribution: RHEL, Rocky Linux, Alma Linux, etc...

> **Note**
>
> The following commands are executed as `root` inside `root`'s home (`/root`). To impersonate root, execute: `sudo -i`

> **Note**
>
> We need to generate some secrets of a minimum given length. Several tools can be used for random. For example, we will use `tr -dc 'A-Za-z0-9'</dev/urandom|head -c 40` to generate a 40 character random string. We could also use `openssl rand -base64 40|head -c 40` for the same.

2.1 Installation parameters

Next sections will use some parameters that you will need to define.

Parameter	Description
`aspera_cert_email`	Place your email, this is used by `letsencrypt` to notify you when the certificate will expire.
`aspera_fqdn`	Place your server's DNS address. For example, I used IBM Techzone and FreeDNS: `itzvsi-f0pjbk8h.mojok.org`
`aspera_rpm`	Path to the HSTS RPM that you downloaded, e.g. `./ibm-aspera-hsts-4.4.5.1646-linux-`
`aspera_eval_lic`	Path to the Aspera HSTS license file, e.g. `./87650-AsperaEnterprise-unlim.eval`
`aspera_os_user`	Typically `xfer`. The operating system user under which transfers will be executed.
`aspera_home`	The home folder of the transfer user. Typically: `/home/xfer`
`aspera_storage_root`	The top folder under which Aspera will transfer files.
`aspera_node_local_port`	The local port where `asperanoded` listens.

Parameter	Description
`aspera_node_local_secu`	`s` for HTTPS, and empty for HTTP. It refers to the local port listened by `asperanoded`.
`aspera_node_user`	The main administrative API user who will create access keys.
`aspera_node_pass`	Password for the latter.
`aspera_https_ext_port`	The external port on which HTTPS will be reachable. Typically, `443`.
`aspera_node_ext_url`	The URL where Node API is accessible.

For convenience, let's create a shell config file `./aspera_vars.sh` with parameters used (assuming to be `root` in `/root`). Execute the following commands in a terminal:

```
test $(id -u) = 0 || echo "ERROR: execute as root"
aspera_rpm=_path_to_hsts_rpm_
aspera_eval_lic=_path_to_license_file_
aspera_cert_email=_your_email_here_
aspera_fqdn=_your_server_fqdn_here_
aspera_os_user=xfer
aspera_home=/home/$aspera_os_user
aspera_storage_root=$aspera_home/aoc
aspera_node_user=node_admin
aspera_node_pass=$(tr -dc 'A-Za-z0-9'</dev/urandom|head -c 40)
aspera_node_local_addr=127.0.0.1
aspera_node_local_port=9092
aspera_node_local_secu=s
aspera_node_local_url=http$aspera_node_local_secu://$aspera_node_local_addr:$aspera_node_local_port
aspera_https_ext_port=443
aspera_node_ext_url=https://$aspera_fqdn:$aspera_https_ext_port
aspera_htgw_local_port=7443
set|grep ^aspera_ > ./aspera_vars.sh
echo 'PATH=/opt/aspera/bin:/usr/local/bin:$PATH' >> ./aspera_vars.sh
```

Once created, edit the generated file `./aspera_vars.sh` and customize with your own values.

```
vi ./aspera_vars.sh
```

Once modified, reload the values:

```
source ./aspera_vars.sh
```

> **Tip**
>
> At any time, if you open a new terminal, you can reload the configuration variables with above command. If you like, you may set the `PATH` in your shell profile as above.

2.2 General system settings

Check that the system has date synchronization:

```
timedatectl
```

If not, then install time synchronization (e.g. `chrony`) and set timezone according to your preference.

```
dnf install -y chrony
systemctl enable --now chronyd
timedatectl set-timezone Europe/Paris
```

Make sure that SELinux is disabled: execute:

```
sestatus|grep mode:
```

```
Current mode:                   permissive
```

If mode is `enforcing`:

- Changes the current operation mode, execute:

```
setenforce Permissive
```

- Change the mode at system startup, execute:

```
sed -i 's/^SELINUX=.*/SELINUX=permissive/' /etc/selinux/config
```

> **Tip**
>
> One can check again with `sestatus`

2.3 Install the Aspera CLI

> **Note**
>
> Installation of the Aspera CLI is not mandatory but simply convenient. It can be installed locally, or on a remote system (Windows, macOS, ...)

User Manual: https://github.com/IBM/aspera-cli

```
dnf module -y reset ruby
dnf module -y enable ruby:3.3
dnf install -y ruby-devel
gem install aspera-cli -v 4.20.0
```

Check installation with:

```
ascli -v
```

2.4 Install the HSTS software

```
dnf install -y $aspera_rpm
```

> **Note**
>
> `perl` is still required by the HSTS installer and also later by Nginx.

2.5 Install the license file

It goes to `/opt/aspera/etc/aspera-license`. This file must be world-readable, or at least readable by `asperadaemons` and transfer users (`xfer`).

```
cp $aspera_eval_lic /opt/aspera/etc/aspera-license
chmod a+r /opt/aspera/etc/aspera-license
```

2.6 Declare the Aspera shell

> **Note**
>
> Optional, good practice, removes some warnings.

As Aspera uses SSH by default, a protection is provided with a secure shell: `aspshell`. This shell can be declared as legitimate shell to avoid warning messages (optional):

```
grep -qxF '/bin/aspshell' /etc/shells || (echo '/bin/aspshell' >> /etc/shells)
```

2.7 Aspera logs

> **Note**
>
> Optional but it is convenient. Aspera logs use syslog and facility `local2`. By default, logs go to `/var/log/messages` with `rsyslog`.

Configure logging per process for Aspera.

```
sed -i -Ee 's/(;cron.none)(\s+\/var\/log\/messages)/\1;local2.none\2/' /etc/rsyslog.conf
echo 'local2.* -/var/log/aspera.log' > /etc/rsyslog.d/99aspera_log.conf
cat << EOF > /etc/logrotate.d/aspera
/var/log/aspera.log
{
  rotate 5
  weekly
  postrotate
    /usr/bin/killall -HUP rsyslogd
  endscript
}
EOF
for d in asperanoded asperaredisd asperacentral asperawatchd asperawatchfolderd asperarund
 ↪  asperahttpd http-gateway ascli async faspio-gateway;do
  l=/var/log/${d}.log
  echo 'if $programname == '"'$d'"' then { action(type="omfile" file="'${l}'") stop }' >
   ↪  /etc/rsyslog.d/00${d}_log.conf
  sed -i -e '/aspera.log/ a '${l} /etc/logrotate.d/aspera
done
systemctl restart rsyslog
```

2.8 Create transfer user

When used with Aspera on Cloud, all transfers are executed under a single technical user (transfer user): `xfer`, specified by `$aspera_os_user`. Optionally we can create a group `asperausers` in case we need to manage multiple transfer users. We make sure to block password-based login with that user and ensure it never expires.

Create this user:

```
groupadd asperausers
useradd --create-home --no-user-group --gid asperausers --shell /bin/aspshell $aspera_os_user
passwd --lock $aspera_os_user
chage --mindays 0 --maxdays 99999 --inactive -1 --expiredate -1 $aspera_os_user
```

2.9 Define storage location root

Let's create some main storage location that will be used by Aspera and make it accessible by the transfer user:

```
mkdir -p $aspera_storage_root
chown $aspera_os_user: $aspera_storage_root
```

2.10 Configure token encryption key

When using Aspera Transfer Token, those are encrypted with a symmetric key. It needs to be provisioned, either as a static key in `aspera.conf` or as a dynamic key in Redis.

2.10.0.1 Static token encryption key

For a PoC, it can be easier to use a static token encryption key:

```
asconfigurator -x 'set_node_data;token_dynamic_key,false'
asconfigurator -x "set_node_data;token_encryption_key,$(tr -dc 'A-Za-z0-9'</dev/urandom|head -c 40)"
```

2.10.0.2 Dynamic token encryption key

> **Note**
>
> Skip this part if you like simplicity and used the static key above.

Refer to the HSTS Documentation.

If you prefer to use dynamic keys:

- First, initialize the global secret (requires a 32 bytes key, base64 encoded): using either `openssl rand -base64 32` or `head -c 32 /dev/urandom|base64` depending on your system:

```
openssl rand -base64 32|askmscli --set-secret-by-category=redis-primary-key
```

> **Note**
>
> This command is done only once, and creates the SQLite DB file `/opt/aspera/etc/rootkeystore.db` . One can peek in this file with: `sqlite3 /Library/Aspera/etc/rootkeystore.db .dump`

- Then, set the key for the transfer user:

```
askmscli --init-keystore --user=$aspera_os_user
```

> **Note**
>
> This command creates the SQLite DB file `~xfer/.aspera/localkeystore.db` with a copy of the primary key.

- Finally, enable dynamic token encryption key:

```
asconfigurator -x 'set_node_data;token_dynamic_key,true'
asconfigurator -x 'set_node_data;token_encryption_key,AS_NULL'
```

2.11 Configure the transfer user for use with tokens

When transfers are authorized with tokens (Aspera Transfer Token or Bearer token, or even Basic token) and if SSH transport is used, then the transfer user must be configured to use public key authentication with Aspera's bypass key.

```
mkdir -p $aspera_home/.ssh
cp /opt/aspera/var/aspera_tokenauth_id_rsa.pub $aspera_home/.ssh/authorized_keys
chmod -R go-rwx $aspera_home/.ssh
chown -R $aspera_os_user: $aspera_home
```

2.12 Other configuration for AoC

Aspera on Cloud requires activity logging:

```
asconfigurator -x
↪  'set_server_data;activity_logging,true;activity_event_logging,true;activity_file_event_logging,
↪  true;activity_bandwidth_logging,true;files_recursive_counts_workers,5'
asconfigurator -x 'set_node_data;pre_calculate_job_size,yes;async_activity_logging,
↪  true;partial_file_suffix,.inprogress'
```

By default, the HSTS uses caching for folder contents. To deactivate folder content caching, execute (Optional):

```
asconfigurator -x 'set_server_data;files_cache_ttl,0'
```

Folder caching is useful when reading folder content is slow, due to slow storage or large number of files in folders.

2.13 Node API user

In order to access the API of HSTS, so we can create an access key, we have to provision an API user:

```
/opt/aspera/bin/asnodeadmin -a -u $aspera_node_user -p $aspera_node_pass -x $aspera_os_user
```

Access keys created with this API user will enable transfers that will be running on the host under user `$aspera_os_user`.

In order to be able to create access keys, we have to remove any docroot and define storage restrictions, to which access key creation will be limited to, for the transfer user. The simplest is to define a loose restriction:

```
asconfigurator -x "set_user_data;user_name,$aspera_os_user;absolute,AS_NULL;file_restriction,|*"
```

Use of token is mandatory, so we need to enable it for the transfer user:

```
asconfigurator -x "set_user_data;user_name,$aspera_os_user;authorization_transfer_in_value,
↪  token;authorization_transfer_out_value,token"
```

When parameters for `asperanoded` (Node API server) are modified, one shall restart the daemon to reload the configuration:

```
systemctl restart asperanoded
```

> **Note**
>
> Similar effect can be achieved with `asnodeadmin --reload`. In case of installation, one can just restart the daemon for config reload.

2.14 Transfer user file restrictions

> **Note**
>
> This section is informational, you can skip to the next section if you are not interested in details.

Skip to next section, if unsure.

The transfer user is associated to a list of file restrictions. Also, the `docroot` shall not be defined for that transfer user. A restriction is a glob (i.e. pattern, not a regex).

Aspera glob syntax is as follows:

- `?` match any single character
- `*` match any number of any character
- `\` escapes the next character (to protect evaluation of one of the special characters: `?*\`)
- any other character is compared as-is

> **Note**
>
> Aspera glob match bytes (8-bit) and does not consider any multibyte encoding (such as UTF8). UTF8 match should work.

For example, for a restriction: `file:////data/*` :

Path	Match?
`file:////data/`	yes
`file:////mnt/`	no
`file:////data/folder`	yes

The syntax of declaration of that list in `asconfigurator` is: `[character][item1][character][item2]...`. The leading character can be anything, and is used as separator later. Typically, `|` is used.

If we want to restrict creation of access keys to only folders under the selected storage location: `$aspera_storage_root`, then one can do:

```
asconfigurator -x "set_user_data;user_name,$aspera_os_user;absolute,AS_NULL;file_restriction,|file:
↪  ///$aspera_storage_root/*"
```

Internally, in HSTS, storage locations are stored as a URI. I.e. `[scheme]://[storage server+credential]/[path]?[pa` . For local storage, `[scheme]` is `file` , and the absolute path starts with `/` . For example, for a local storage `/data` , the URL would be `file:///data` .

At the time of creation of access key, the access key storage root URI will be validated against the list of restriction globs. Note that an access key is created with a storage location specified as JSON, not URI. Aspera internally converts the storage location to a URI, and then checks that the URI matches one of the globs in the restriction list. If the restriction list is only `file:///data` (no glob), then only that precise path will be allowed. Else, in order to allow any path under two locations: `/data/mnt1` and also S3 storage `s3://mys3/bucket` , the restriction list would be `file:///data/mnt1/*` and `s3://mys3/bucket/*` , and command would be:

```
asconfigurator -x "set_user_data;user_name,$aspera_os_user;absolute,AS_NULL;file_restriction,|file:
↪  ///data/mnt1/*|s3://mys3/bucket/*"
```

Note

The restriction list does not define the storage location, it is a protection to limit the creation of access keys to only some locations.

2.15 SSH server configuration

By default, Aspera uses SSH for Aspera transfer session initiation. It is also possible to configure HTTPS for token-based authorization. As recommended by IBM, do not expose port 22, and prefer to use port `33001` for SSH connections for Aspera. One can either use a single SSH server (`sshd`) for both remote terminal and Aspera transfers, or use a separate SSH server for Aspera transfers.

2.15.0.1 Common SSH server for remote access and Aspera transfers

This is the simplest configuration, as one only needs to configure the SSH server to listen on port `33001` instead of `22` .

Let's configure SSH to also listen on port 33001 only:

```
sed -i '/^#Port 22$/a Port 33001' /etc/ssh/sshd_config
sed -i '/^#UseDNS yes$/a UseDNS no' /etc/ssh/sshd_config
sed -i '/^HostKey .*ecdsa_key$/s/^/#/ ' /etc/ssh/sshd_config
sed -i '/^HostKey .*ed25519_key$/s/^/#/ ' /etc/ssh/sshd_config
systemctl restart sshd
```

Note

To keep both 33001 and 22, uncomment the line: `#Port 22` , then restart the SSH service.

2.15.0.2 Separate SSH server for Aspera transfers

It is possible to spawn a totally separate SSH server for Aspera transfers. This allows to keep the default SSH server for remote access, and to use a separate SSH server for Aspera transfers with a different configuration (and port).

TODO

2.16 Public IP and DNS

In order to work with Aspera on Cloud, it is required to have a public IP address on which the following ports are open:

Port	Usage
TCP/33001	FASP Session (SSH)
UDP/33001	FASP Data
TCP/443	Node API (HTTPS)
TCP/80	Optional: if `letsencrypt` is used

Once the DNS name is known:

```
echo $aspera_fqdn > /etc/hostname
hostname $aspera_fqdn
hostname
```

2.17 Certificate with let's encrypt

A TLS certificate is required for above FQDN.

If you don't have one, then it is possible to generate one with below procedure using `letsencrypt` :

Install `certbot` :

```
dnf install -y python3.12
python3 -m venv /opt/certbot/
/opt/certbot/bin/pip install --upgrade pip
/opt/certbot/bin/pip install certbot
ln -s /opt/certbot/bin/certbot /usr/bin/certbot
```

Generate a certificate:

```
certbot certonly --agree-tos --email $aspera_cert_email --domain $aspera_fqdn --non-interactive
 ↪  --standalone
```

Note

For above command to work, the FQDN shall resolve in DNS and port TCP/443 reachable. Certificate and key is placed here: `/etc/letsencrypt/live/$aspera_fqdn/` , see Let's encrypt documentation

2.18 Nginx

Technically, Nginx is not required, but it is recommended when the Node API fronts Internet, and it has several advantages. It :

- allows using port 443 for HTTPS, as `asperanoded` runs as user `asperadaemon` and cannot bind to port `443` ,
- simplifies the installation of certificates,
- adds a security layer with a well-known reverse proxy,
- allows to use a single port for both Node API and transfers (WSS, if configured) and other services.

Since we will use Nginx as reverse proxy, we can make Node API listen locally only:

```
asconfigurator -x
 ↪  "set_server_data;listen,$aspera_node_local_addr:$aspera_node_local_port$aspera_node_local_secu"
systemctl restart asperanoded
```

Note

`s` is for HTTPS. Restart is required to change listening address.

Install Nginx:

```
dnf install -y nginx
```

Create a configuration file for Nginx:

- This one uses the `letsencrypt` certificate. If you used another method, then reference the actual location of the certificate and key in parameters `ssl_certificate*`

```
cert_chain_file=/etc/letsencrypt/live/$aspera_fqdn/fullchain.pem
cert_key_file=/etc/letsencrypt/live/$aspera_fqdn/privkey.pem
cat<<EOF > /etc/nginx/conf.d/aspera.conf
server {
  listen                    $aspera_https_ext_port ssl;
  listen                    [::]:$aspera_https_ext_port ssl;
  server_name               _;
  root                      /usr/share/nginx/html;
  ssl_certificate           $cert_chain_file;
```

```
  ssl_certificate_key        $cert_key_file;
  ssl_session_cache          builtin:1000 shared:SSL:10m;
  ssl_protocols              TLSv1.2 TLSv1.3;
  ssl_ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:
↪  !aNULL:!MD5:!DSS;
  ssl_prefer_server_ciphers  on;
  access_log                 /var/log/nginx/global.access.log;
  proxy_set_header           Host              \$host;
  proxy_set_header           X-Real-IP         \$remote_addr;
  proxy_set_header           X-Forwarded-For   \$proxy_add_x_forwarded_for;
  proxy_set_header           X-Forwarded-Proto \$scheme;
  proxy_read_timeout         90;
  proxy_buffering            off;
  proxy_request_buffering    off;
  server_tokens              off;
  # HSTS: node API
  location / {
    proxy_pass               $aspera_node_local_url;
    proxy_hide_header        Access-Control-Allow-Origin;
    add_header               Access-Control-Allow-Origin *;
    access_log               /var/log/nginx/node.access.log;
  }
  # HTTP Gateway
  location /aspera/http-gwy {
    proxy_pass                https://127.0.0.1:$aspera_htgw_local_port;
    access_log                /var/log/nginx/httpgw.access.log;
    proxy_http_version        1.1;
    proxy_set_header          Upgrade \$http_upgrade;
    proxy_set_header          Connection "Upgrade";
    proxy_set_header          Host \$host;
  }
}
EOF
```

Note

If a reverse HTTP proxy in from of the Node API, with a different port, then include both ports in the config file above.

Then start and enable it permanently (start on reboot):

```
systemctl enable --now nginx
```

2.19 Verification

Note

Ideally, below command shall be executed from outside the on-premise environment. The goal being to verify that Aspera on Cloud services can correctly access the on-premise server and that the certificate is well recognized from internet.

At this point, Nginx shall be forward requests to the Node API and an API user and transfer user shall be configured.

Check with:

```
curl -u $aspera_node_user:$aspera_node_pass $aspera_node_ext_url/info
```

Check that the following values are set like this:

```
"transfer_user" : "xfer",
"docroot" : "",
```

2.20 Creation of access key and node using AoC web UI

In the Aspera on Cloud web UI, navigate to `Admin app` → `Nodes and storage` → `Create new +`

- Select tab: `Attach my Aspera server`
- Name: anything you like to identify this node by name
- URL: value of: `$aspera_node_ext_url`
- Leave other as default
- Select radio button `Create a new access key`
- Node username: `$aspera_node_user`
- Node password: `$aspera_node_pass`
- Storage: `Local Storage`
- Path: `$aspera_storage_root`

> **Note**
>
> The Path used for access key creation must pass glob validation with the restriction list created earlier. If the glob was ending with a `*`, then the Path can be any folder below the folder prefix.

2.21 Creation of access key and node using `ascli`

Here, we are going to create the access key using the CLI, which uses the node API.

2.22 Configure `ascli`

Configure access to Node API:

```
ascli config preset update node_admin --url=$aspera_node_ext_url --username=$aspera_node_user
 ↪  --password=$aspera_node_pass
ascli config preset set default node node_admin
```

2.23 Create the access key

```
ascli node access_keys create @json:'{"storage":{"type":"local","path":"'$aspera_storage_root'"}}'
 ↪  --show-secrets=yes|tee my_ak.txt
```

The access key credentials are displayed and saved in file: `my_ak.txt`

2.24 Create the node

In the Aspera on Cloud web UI, navigate to `Admin app` → `Nodes and storage` → `Create new +`

- Select tab: `Attach my Aspera server`
- Name: anything you like to identify this node by name
- URL: value of: `$aspera_node_ext_url`
- Leave other as default
- Select radio button `Use existing`
- Access key: value from `my_ak.txt`
- Secret: value from `my_ak.txt`

2.25 Accessing AoC using command line

Configure access to Aspera on Cloud: `myorg` is the name of the AoC tenancy (organization), i.e. the first part of the address of the URL. One can also place the URL of the org: `https://myorg.ibmaspera.com`

```
ascli config wizard [myorg] aoc
```

Then follow the Wizard.

> **Note**
>
> When using the CLI, a user will be authenticated using a private key. AoC supports a single public key per user. If the user uses the CLI from multiple systems, then the same private key shall be used on those systems (for example on the Aspera Transfer Server, and on a laptop).

2.26 Configure Aspera Event Journal Daemon (AEJD)

The Aspera Event Journal Daemon is responsible to report events from the Aspera Transfer Server, back to the Aspera on Cloud API. It reports file events (transfers, etc...).

2.27 Special case: HSTE

If the transfer server is an HSTS, skip this step.

If the node is an Aspera Endpoint, then create this file: `/opt/aspera/etc/systemd/asperaejd.service` with this content:

```
[Unit]
Description=IBM Aspera Event Journal Daemon
ConditionPathExists=/opt/aspera/sbin/aejd
StartLimitInterval=0

[Service]
User=asperadaemon
Group=aspadmins
Type=simple
PIDFile=/opt/aspera/var/run/aspera/aejd.pid
ExecStart=/opt/aspera/sbin/aejd
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutStopSec=20
KillMode=process
Restart=always
RestartSec=10s
```

Then activate AEJD. Execute as root:

```
/opt/aspera/etc/setup/setup-systemd.sh enable
```

The AEJ Daemon shall now be known. Its status can be shown with:

```
systemctl status asperaejd
```

2.28 Create a node registration token

This token can be used a single time. It can be created using the AoC web UI, or using `ascli` (requires to have configured access to AoC through `ascli`, see previous section):

This command saves the generated token in shell variable: `$registration_token`

```
registration_token=$(ascli aoc admin client_registration_token create @json:'{"data":{"name":
↪  "laurentnode","client_subject_scopes":["aejd"],"client_subject_enabled":true}}' --fields=token
↪  --show-secrets=yes)
```

To display the value:

```
echo $registration_token
```

This value will be used only once.

2.29 Activate the AEJ Daemon

Execute as `root` (Still assuming that `/opt/aspera/bin/` is in the `PATH`):

This command activate reporting of events from Node Daemon to the AEJ Daemon, once Node Daemon is restarted.

```
asconfigurator -x "set_server_data;aej_logging,true;aej_port,28000;aej_host,$aspera_node_local_addr"
```

Use the token from previous step in: `registration_token` variable. This command creates the configuration file: `/opt/aspera/etc/aejd.conf` after calling back AoC API to register the node.

```
/opt/aspera/bin/asp-cloud-config tether --aoc-registration-token $registration_token --aoc-url
 ↪  https://api.ibmaspera.com
chmod 600 /opt/aspera/etc/aejd.json
chown asperadaemon: /opt/aspera/etc/aejd.json
```

Note

As of 4.4.5 HSTS, the command `asp-cloud-config` has a defect where the config file `aejd.conf` is created in `$PWD/../etc` instead of `/opt/aspera/etc` if the command is executed without a full path. So either move to `/opt/aspera/bin/` before executing, or use the full path to the command like proposed here. Also, the resulting file `aejd.json` shall be readable by user `asperadaemon` .

Restart Aspera services in that order to apply the configuration:

```
systemctl restart asperaejd
systemctl restart asperanoded
```

2.30 Installation of HTTP Gateway

The HTTP Gateway can be installed on the same server as the Aspera Transfer Server.

2.30.1 Installation

```
rpm -Uvh ibm-aspera-httpgateway-2.3.0.156-b3b9633.x86_64.rpm
```

2.30.2 Configuration

Make HTTP Gateway listen locally on high port.

```
cp /opt/aspera/httpgateway/config/{default,gatewayconfig}.properties
sed --in-place --regexp-extended \
--expression='s/^(serverconfig\.host)=.*/\1=127.0.0.1/' \
--expression="s/^(serverconfig\.port)=.*/\1=$aspera_htgw_local_port/" \
/opt/aspera/httpgateway/config/gatewayconfig.properties
systemctl restart aspera_httpgateway
```

Auto restart on failure:

```
service_file=/usr/lib/systemd/system/aspera_httpgateway.service
if ! grep -q '^Restart=' $service_file;then
  tmpfile=$(mktemp)
  cat > "$tmpfile" << EOF
Restart=on-failure
RestartSec=30s
EOF
  sed -i -Ee '/^\[Service\]/r '"$tmpfile"'' $service_file
  rm -f "$tmpfile"
  systemctl daemon-reload
  systemctl restart aspera_httpgateway
fi
```

2.30.3 NGINX Configuration

NGINX is configured as reverse proxy for HTTP Gateway. See section NGINX Configuration for details.

Chapter 3

Maintenance operations

3.1 Transfer server backup

Some configuration of the Transfer server can be re-created easily, such as node AI user, static configuration (`aspera.conf`) or even access keys.

But some other state information cannot be re-created, as it is the result of file transfers. Such information include file identifiers and permissions. Those are stored in a local database. So it is important to proceed to a regular backup of this information.

In case of disaster, the Aspera transfer Server node shall be rebuilt. This includes:

- installation and configuration of Operating system
- installation and configuration of Aspera Software
- installation and configuration of other Software (Nginx)
- restoration of state backup

An easy way to prevent disaster, in the case of use of Virtual Machines, is to perform a snapshot of the storage.

The installation and configuration of software can even be automated using tools such as Red Hat Ansible and IBM HashiCorp Terraform.

3.1.1 System files

Any customization to the OS must be restored, such as the ones listed in this document.

3.1.2 HSTS config files

The following files shall be backed up:

- `/opt/aspera/etc/aspera.conf`
- `/opt/aspera/etc/aspera-license`
- `/opt/aspera/etc/conf.d/node_id.conf`
- `/opt/aspera/etc/conf.d/cluster_id.conf`

3.1.3 HSTS Redis DB

Refer to the HSTS documentation for details on backup and restore of the HSTS Redis database, section: `Backing up and restoring a node database` .

3.2 Changing FQDN and certificate

If the hostname (FQDN) of the HSTS needs to be modified, the associated certificate also needs an update.

Prerequisites:

- Get a certificate for that new FQDN

- Register this FQDN in DNS (A or AAAA record)
- For convenience, edit the file `aspera_vars.sh` and update the value for `aspera_fqdn`.

```
sed -i.bak -E -e "s|^(aspera_fqdn=).*|\1newhost.example.com|" ./aspera_vars.sh
source ./aspera_vars.sh
set|grep ^aspera_
```

3.2.1 Updating local hostname

```
echo $aspera_fqdn > /etc/hostname
hostname $aspera_fqdn
```

Check with:

```
hostname
```

```
newhost.example.com
```

Edit the file: `/etc/hosts`, and, at the end of the line with `127.0.0.1`, add that FQDN:

```
127.0.0.1 localhost newhost.example.com
```

Alternatively:

```
echo "127.0.0.1 $aspera_fqdn" >> /etc/hosts
```

Check with (or with `ping`):

```
getent hosts $aspera_fqdn
```

```
127.0.0.1 localhost newhost.example.com
```

> **Note**
>
> This entry in `/etc/hosts` is used in case of a local HSTS transfer, in AoC that is the case for a move or a copy.

3.2.2 Storing the certificate and private key

The certificate chain and its key should be stored in a location accessible by Nginx. It can be anywhere, including a standard location:

```
openssl version -d
```

```
OPENSSLDIR: "/etc/pki/tls"
```

Let's store certificate files in standard locations:

- `/etc/pki/tls/certs/newhost.example.com.fullchain.pem`
- `/etc/pki/tls/private/newhost.example.com.key.pem`

Let's adjust access rights: By default, Nginx runs as user `nginx`

```
eval $(openssl version -d|sed 's/: /=/')
cert_chain_file=$OPENSSLDIR/certs/$aspera_fqdn.fullchain.pem
cert_key_file=$OPENSSLDIR/private/$aspera_fqdn.key.pem
chmod 644 $cert_chain_file
chmod 600 $cert_key_file
chown nginx: $cert_key_file
```

> **Note**
>
> The certificate file should contain the full chain.

3.2.3 Configuration for Nginx

Refer to the Nginx documentation.

Modify `/etc/nginx/nginx.conf`, and change parameters: `ssl_certificate` and `ssl_certificate_key` with above paths.

```
sed -i.bak -E -e "s|(ssl_certificate\s+).*;|\1$cert_chain_file;|" /etc/nginx/nginx.conf
sed -i.bak -E -e "s|(ssl_certificate_key\s+).*;|\1$cert_key_file;|" /etc/nginx/nginx.conf
```

```
systemctl restart nginx
systemctl status nginx
```

Check with:

```
curl -i $aspera_node_ext_url/ping
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 05 May 2025 14:11:27 GMT
Transfer-Encoding: chunked
Connection: keep-alive
```

3.2.4 Change the node URL in AoC

This can be done using the web UI: Admin → Nodes and storage → Nodes → Profile or the CLI as below.

First identify the node identifier that you configured:

```
ascli aoc admin node list
```

Either use the numerical identifier `_my_node_id_` , or, if you know the name: `%name:"my node name"`

```
ascli aoc admin node modify _my_node_id_ @json:'{"url":"'$aspera_node_ext_url'"}'
```

End of document