FastX 3.2 Gateway Guide

For FastX 3.2

Background

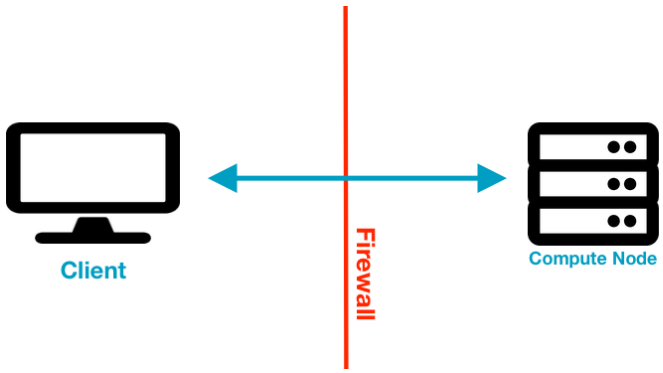
Many users wish to separate their compute nodes from the outside world. This separation offers better security and allows administrators to better monitor the traffic between clients and servers. There are multiple ways to create this separation. For example:

- VPN
- Reverse Proxy
- FastX Gateway

Each method offers its advantages and in most cases, a specific FastX gateway offers no advantage of the traditional methods.

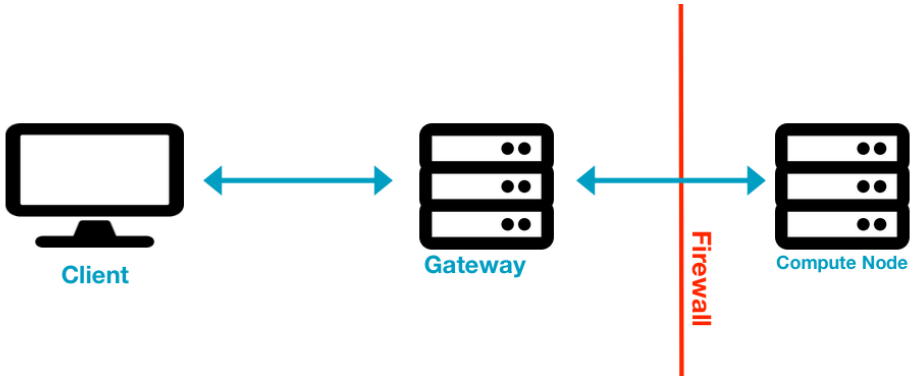
VPN Method

The VPN method is the most straightforward method. Configure the network such that the clients have direct access to each compute node in your cluster. In this instance no extra work is needed. Gateways are not needed at all



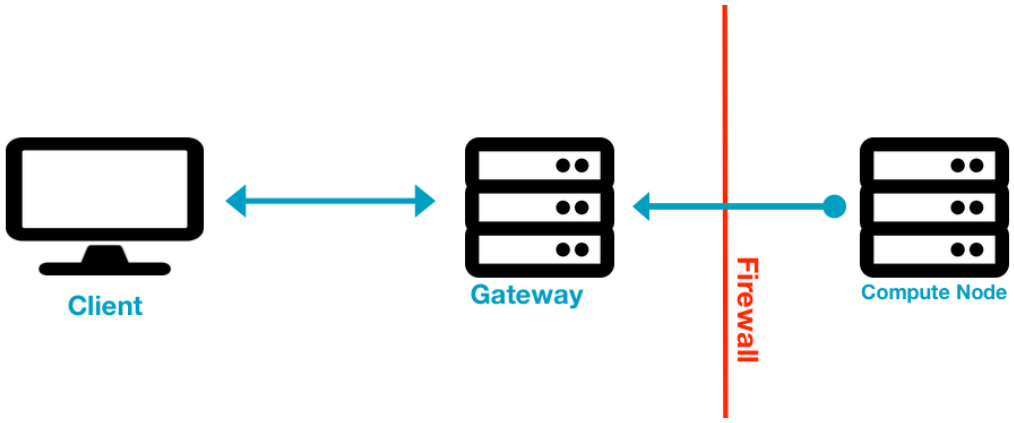
Reverse Proxy

The reverse proxy method installs a proxy server on the edge of the network. The compute nodes in the cluster are isolated from the outside networks. The gateway server can still directly access the compute nodes (for example a passthrough in the firewall, or multiple network cards). In this scenario it is best to use a traditional proxy server like Apache or nginx.



FastX Gateway

The FastX Gateway method is used when the compute nodes are completely isolated from the outside network. There is no way a proxy server can connect in to the isolated network. The compute nodes, however, can connect out to an external server that can act as a gateway. The compute nodes set up tunnels to the gateway server. Only then can the gateway proxy messages to the compute nodes.

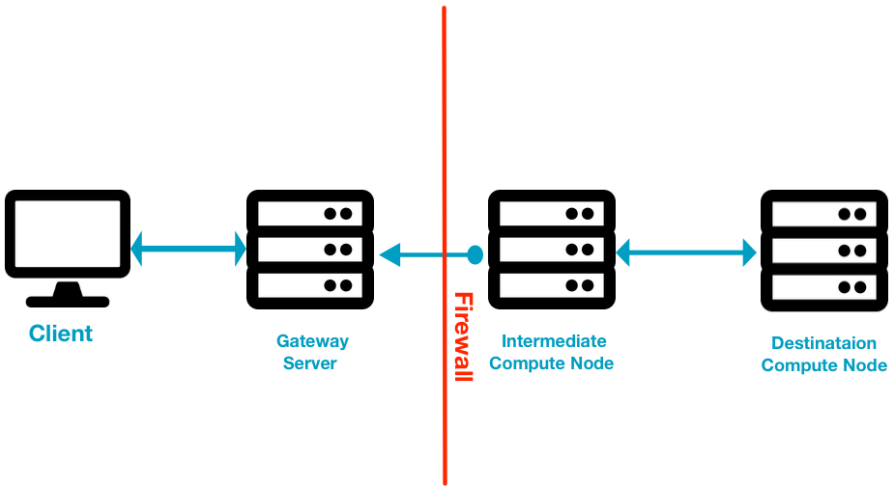


The VPN and Reverse Proxy method are outside of the scope of this guide. These methods have been used for many years and are well documented. The remainder of this guide deals with the FastX Gateway configuration.

Relationship to FastX Cluster

The Gateway Server is not a FastX Server. Users cannot launch FastX on a Gateway Server without first installing the FastX rpm.

FastX cluster members must be able to directly connect to other cluster members. This typically means that all cluster members are installed on the same LAN.



The Gateway Server now acts as a tunnel from the external system to one of the Compute Nodes of the FastX cluster. This is a many Compute Nodes to one Gateway Server relationship. (You can set up multiple Gateway Servers, but each Compute Node only connects to one specific Gateway Server).

Load Balancing

Load Balancing is only done on cluster members, not on the Gateway Server. When an admin sets up a load balancing script, the script is executed by a Compute Node, and then the Compute Node will proxy the request to the destination server.

Installation

Gateway Installation

Online Installation

These instructions are for gateways which have access to the public internet. The installation will automatically download and configure all the necessary applications from standard repositories. Please use the offline installation instructions if your gateway does not have access to the public internet, or if you wish to use your own repositories.

Compute Node Installation

These instructions are optional for administrators who want to use a gateway configuration. You do not need to set up every compute node to connect to a gateway server. You can specify a subset of compute nodes that will connect to different gateways. The gateway will proxy messages down to this compute node and then it will relay it to the destination node. Note that this will add extra load on specific compute nodes.

Copy the gateway.json file from the Gateway Server into /usr/lib/fastx/var/config/gateway.json and restart the service

Copy the gateway.json file that was created in the gateway installation and put it in

$FX_CONFIG_DIR/gateway.json

Default: /usr/lib/fastx/var/config/gateway.json

Configuration

Public Configuration

```
{
    "port": 3300,
    "cert_file": "/path/to/SSL_Certificate",
    "key_file": "/path/to/SSL_Private_Key",
    "ca_file": "/path/to/CA_File",
    "https": { /* TLS Options */ },
    "middlewares": [
          "/path/to/m1.js",
          "/path/to/m2.js",
          ...
          "/path/to/mN.js"
    ]
}
```

The public configuration will configure the public facing portion of the Gateway Server. This is the portion that the users will connect to. Admins should use Trusted Third Party Certificates when configuring this portion of the gateway.

Middlewares

Middlewares are a special set of functions that will be run in order before tunneling any request to the Compute Node. Middlewares are nodejs modules of the Express middleware format.

```
/**
  @param {object} req - request object
  @param {object} res - response object
  @param {function} next - next function to continue
*/
module.export = function(req, res, next) {
   /* Your code here */
   next();
}
```

The middleware function allows admins to add any special code into the gateway. This can include

- Logging all requests
- Add extra headers
- Restricting IPs
- Restricting users to some or all of the web api
- Emailing admins when an api is triggered
- Anything an admin wants to do

Several middlewares are shipped with the gateway. They are located in */path/to/gateway/plugins*

Relay Configuration

```
{
    "cert_file": "/path/to/SSL_Certificate",
    "key_file": "/path/to/SSL_Private_Key",
    "ca_file": "/path/to/CA_File",
    "https": { /* TLS Options */ },
    "router": "/path/to/router.js"
}
```

The relay configuration will configure the portion of the Gateway Server that the Compute Nodes connect to. This server can be encrypted using Trusted Third Party Certificates

Router

The router option is a special option that can act as a short circuit to a specific Compute Node. Multiple Compute Nodes can connect to a single Gateway Server. There may be instances where an admin may wish to add some specific routing logic to the Gateway Server based on user defined criteria.

A default router is shipped with the gateway. This will route specific session requests (like connect, terminate etc) to the correct compute node if it is available. Otherwise it will use round robin routing.

```
/**
  @param {object} req - request object
  @param {Map} clientMap - map of all the connected compute nodes
  @return {Client} - connected compute node
*/
module.exports = function(req, clientMap) {
      let cl = client.get("serverId");
      return cl;
}
```

The default router is located in */path/to/gateway/router/router.js*

Gateway Configuration

```
{
    "url": "https://gateway.example.com:33300",
    "port": 33300,
    "host": "0.0.0.0",
    "key": "JWT_SECRET_SHARED_KEY",
    "options": { /* JWT Options */ },
}
```

Notes on Options

The gateway configuration is meant to be easily distributed between gateway servers and compute nodes. Certain options are only used depending on if it is a gateway or a compute node.

- url - url of the gateway server.
    - Compute Node - gateway server to connect to
    - Gateway Server - if port is missing, the gateway will parse the url and bind to the port
- port - port to bind to
    - Compute Node - used to generate a url if url is missing
    - Gateway Server - default port to bind to
- host - bind host on the Gateway server
    - Compute Node - if url is missing, will use host and port to generate a url

    - Gateway Server - bind host
- key - secret shared key used to generate a JWT
- options - JWT options