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:

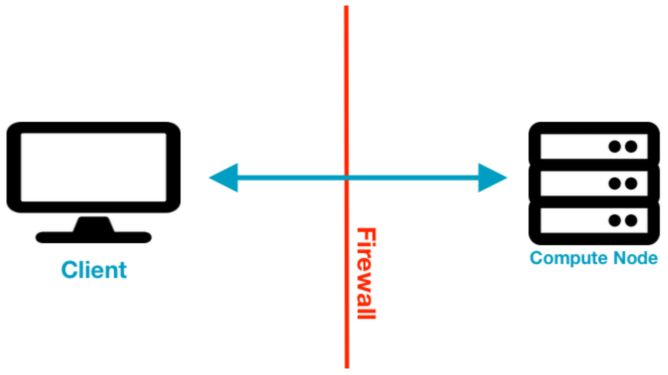
- VPN
- Reverse Proxy
- Gateway

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

Note: A “Compute Node” used in this document is also known as a “FastX Server” or “FastX Cluster Member”.

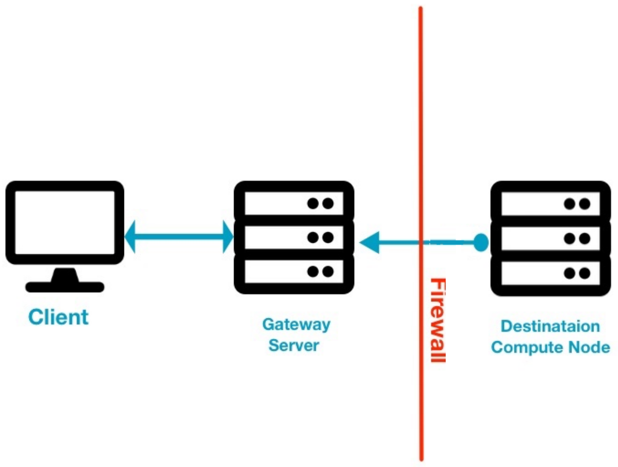
VPN Method

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



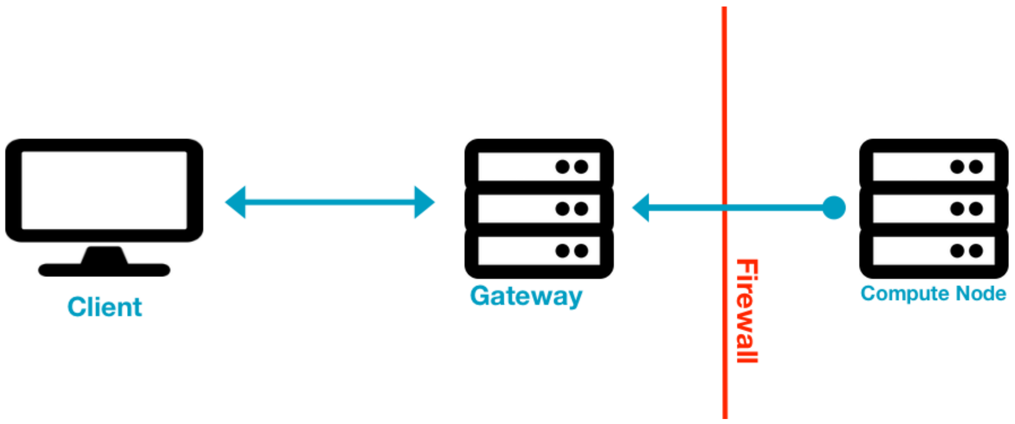
Reverse Proxy

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.



Gateway

The 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. This method requires that the compute nodes are able to make an outbound connection through the firewall to the gateway.

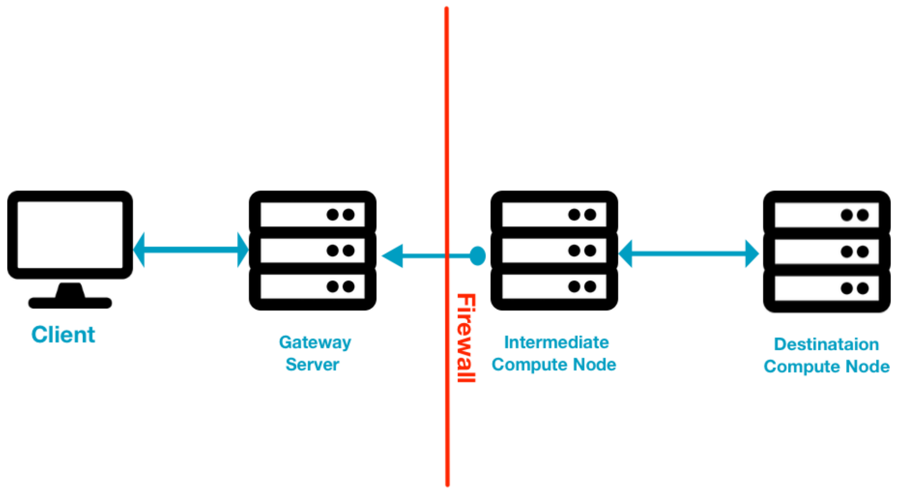


The VPN method are outside of the scope of this guide. These methods have been used for many years and are well documented. The remainder of this guide deals with the FastX Gateway configuration.

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 server.

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 typically 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 Server Installation

Requirements

- SSH server installed

Instructions

The gateway server is shipped via the FastX repository. First, enable the repository by running the command

curl https://www.starnet.com/files/private/Beta/setup-fastx-gateway-preview.sh | sudo bash -

Debian/Ubuntu Systems

Install the gateway server by running the commands

```
sudo apt update
sudo apt install fastx-gateway
```

RedHat/CentOS/RPM Based Systems

```
sudo yum install fastx-gateway
```

Compute Node Installation

The FastX server ships with a tunnel script that will make connections to the gateway server. In a cluster, only one system is required to connect to the gateway. The other cluster members can be reached from this frist compute node However this introduces a single point of failure on your network. We recommend nominating a subset of your compute nodes to connect to the gateway for fault tolerance. You may also connect different cluster members to different gateways. Each gateway will still access the same cluster.

Requirements

- FastX Server already installed
- SSH Client Installed (default on most systems)

Installation

To configure a FastX server to connect to the gateway, run the script
/usr/lib/fastx/3/install/tunnel

This script will prompt for the name or IP of the gateway. The tunnel script creates an SSH connection to the gateway, and connects with an SSH key pair. You need to copy the public key to the gateway machine for access. Follow the instruction in the installation script to add the key.

Configuration

The FastX server uses 2 configuration files: public and proxy. In the default installation, these files are found in

- /etc/fastx/gateway-public.json
- /etc/fastx/gateway-proxy.json

The names of these files can be changed by setting the “FX_PUBLIC_CONFIG” and/or FX_PROXY_CONFIG variables in the /etc/sysconfig/fastx-gateway file.

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 */ },
}
```

The public configuration will configure the public facing portion of the Gateway Server. This uses the same file format as the FastX server's www.json file. If running on the same system as a FastX server, make sure to use a different port number.

Proxy Configuration

The proxy configuration file does the bulk of the operations. It is involved in setting up plugins, adding reverse proxies, adding gateway proxies and routing.

```
{

    "plugins": [ "/path/to/plugin1", "/path/to/plugin2" ],
    "proxies": [ { "target": "https://server.example.com:3300", "secure":
false }],
    "filename": "/etc/fastx/gateway.db",
    "routers": [ "/path/to/router1", "/path/to/router2"]
}
```

Plugins

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

Note on middleware: Your function should either call res.json, res.send, res.sendFIle ... or next as the last line. res is the result object and it will prevent any other middlewares (and proxying to continue). next is the next function and will continue on the path. If you call next with any parameter next(o) it will be interpreted and an HTTP error and send the result.

Proxies

Proxies are typical reverse proxies. These are created at program launch and will stay for the duration of operation.

You can create as many proxies as you want. Typically only the hostname and port are required. For advanced configuration see https://www.npmjs.com/package/http-proxy#options

Creating a proxy does not automatically forward information. This is done in the router portion.

Filename

The filename is a special file that the gateway will use to create dynamic proxies. These proxies are created by the FastX Compute Nodes creating tunnels to the gateway server. Once the tunnel is created, they store the information in a gateway.db file. The Gateway will read this file periodically and create tunnels based on this. This allows access to servers that are typically blocked by a firewall.

NOTE: If all your systems (including FastX Gateway) are on the LAN, you do not need this option. Use"proxies" instead.

Routers

The routers are a series of plugins that execute a routing function. The purpose of a router is to determine which proxy (static or dynamic) to use in order to forward the packets.

The router function has the following format.

```
/**
     @param {object} staticProxies - array of proxies that came from the
proxies object
@param {object} dynamicProxies - array of proxies that were built using the
filename ssh-tunnel
  @param {object} req - request object
  @param {object} res - response object
  @param {function} next - next function to continue
*/
module.export = function(staticProxies, dynamicPoxies, req, res, next) {
   /* Your code here */
   if(/*proxy found */) {
       return { result: staticProxy[i]; }
   } else if(/*I want to filter the proxies for the next router */) {
      return {
       staticProxies: filteredListOfStaticProxies,
       dynamicProxies: filteredListOfDynamicProxies
      }
   }
   return  /* nothing, move on to the next router */
}
```

Routers enable the administrators to create custom logic based on their use case. You may wish to route requests to different servers based on their IP address. Or you may intercept an api request and short circuit it to send it directly to the destination to improve performance. You can basically do anything you want with a router.

Meta object

Both the static and dynamic proxies will have a meta parameter which is the object of the information used to create the proxy. This can be utilized to determine if the proxy should be used. For example, you can add "clusterMember": "abc" as part of the meta object and decide to route to server "abc" from there

Returning a result

Once you have found the proxy (static or dynamic) you wish to use, return it an object with a result parameter. The result is the proxy you are using

```
{ result: myProxy }
```

This tells the gateway that you are done and then the gateway can continue operation

Filtering proxies

If you have multiple routers, you can use the router function to filter out proxies and then continue to the next operation. This can allow you to better organize your routers and reuse them when your needs change.

To filter proxies, return an object with the following optional parameters

```
{
    staticProxies: filteredStaticProxies,
    dynamicProxies: filteredDynamicProxies
}
```

The filtered proxies will be used as the list in the next router

Localhost Gateways

There are many use cases where you may want to install the Gateway on the same system as a FastX server and route only to that server. This can allow you to perform your custom plugin logic before a user accesses FastX. Examples include:

- Log the IP address of an incoming request
- Prevent users from outside localhost from accessing the admin section
- Automatically rejecting specific IP addresses or users
- Any other logic needed to integrate better with your operations

In this scenario, you should simply add a static proxy that goes to *localhost:YOUR_PORT*
Remember, the Gateway Server and the FastX Server are different Http servers, so your ports need to be different in the Gateway's public-config.json and the FastX server's www.json

StarNet Communications Corp.
4677 Old Ironside Drive, Suite 210, Santa Clara, California 95054-1825 • USA
408.739.0881 • 408.739.0936 • sales@starnet.com • www.starnet.com