COMMVAULT®

Configuring a network load-balanced Web Console

Author: Ruben Renders, Solution Architect, Professional Services, Commvault EMEA

April 30, 2020

Table of contents

Introduction

This technical document can be used as general guidance to configure a Web Console stack behind a Network Load Balancer (NLB) for high-availability. The NLB can be configured to perform SSL offloading and can be integrated with Microsoft Active Directory Federation Services (Microsoft ADFS) to support SAML authentication. The integration of the authentication mechanism with an external Identity Provider (Idp) has been excluded from the configuration.

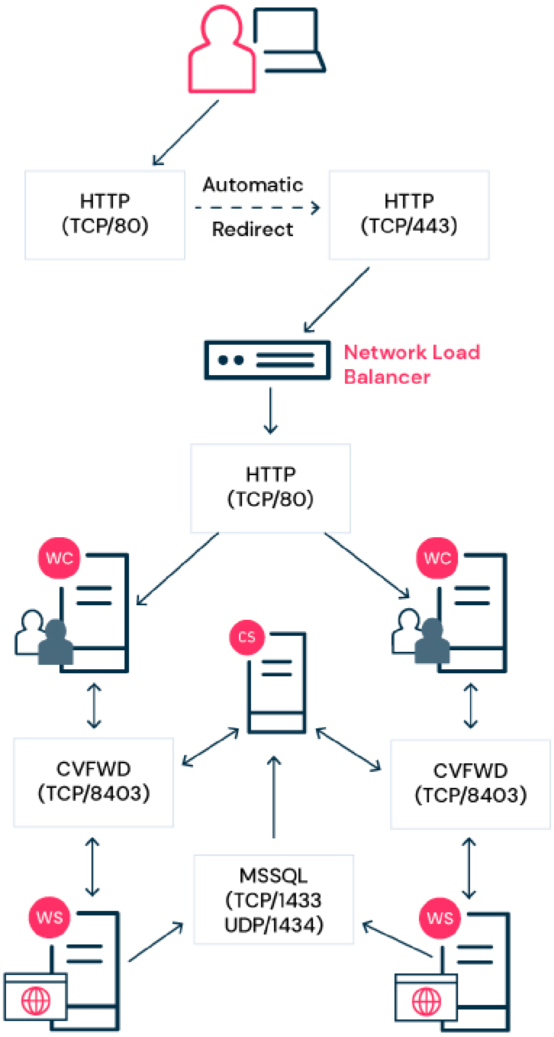
The information outlined in this document has been collected from a current Commvault customer using Commvault as a Managed Service Provider (MSP) deployment. The configuration of Microsoft ADFS and the redundant Web Console stack is part of a service optimization exercise and will be the foundation for the Commvault Command Center adoption.

Environment details:

- The Web Console and Web Server components are installed on the same system.
- The Web Console systems reside behind an F5 BIG-IP Network Load Balancer.
- The NLB is configured to perform SSL offloading (HTTPS > HTTP).
- The integration has been performed on Commvault version 11.19.1.

High-level architecture

The diagram below depicts the overall topology, and all components required for successful end-user communication through a firewall. This environment uses a two-way firewall configuration with a tunnel port (cvfwd - TCP/8403). The arrows indicate if the port needs to be open inbound, outbound, or both.



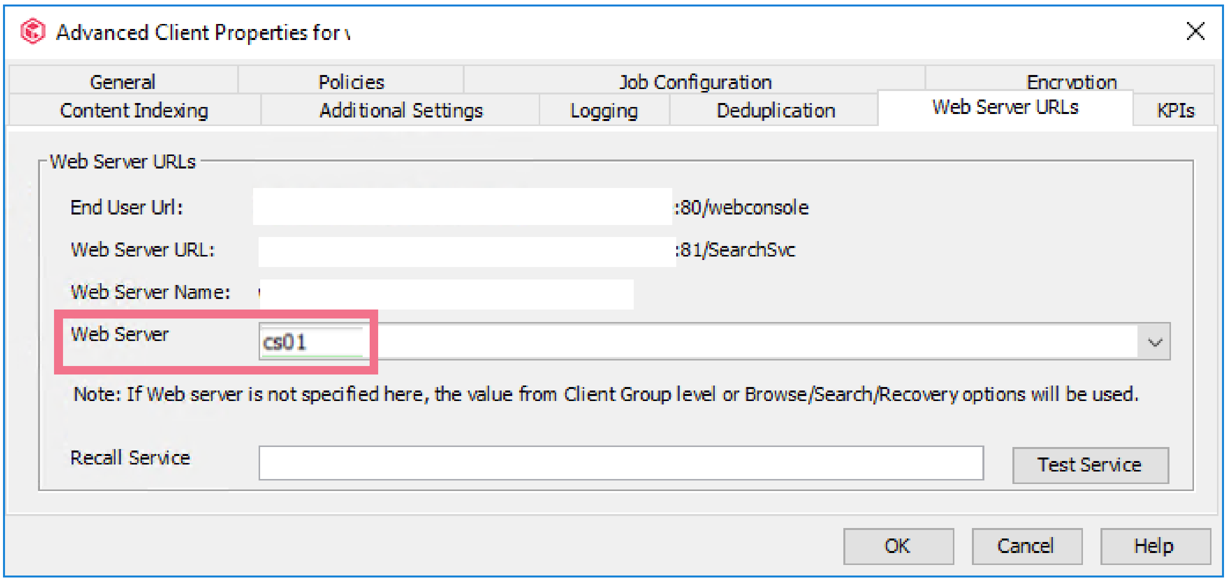
Configuration

Step 1 – Building the web console/webserver stack

Perform a regular installation of at least two systems containing the Web Console - and Web Server role.

- Stack 1: Web Console server 1 using Web Server 1
- Stack 2: Web Console server 2 using Web Server 2

You can verify the Web Server being used on the client running the Web Console. The details can be found in the CommCell Console GUI on the Web Console Client. Access the "*Client Properties* > *Advanced* > *Web Server URLs*".



End User Url: :80/webconsole

Web Server URL: :81/SearchSvc

Web Server Name:

Web Server cs01

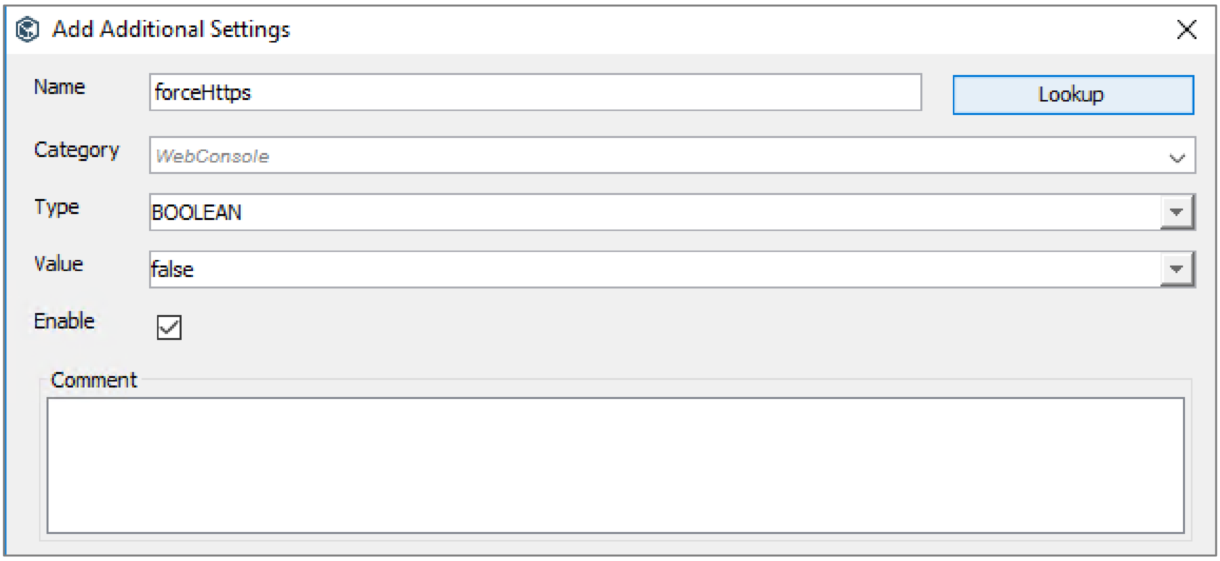
Note: If Web server is not specified here, the value from Client Group level or Browse/Search/Recovery options will be used.

Recall Service

Step 2 - Implement the additional setting "forceHTTPS" with value "false" on the Web Console

As of version 11 feature pack 9, all connections are forced to tunnel through HTTPS. As we are performing SSL offloading, we expect all connections to come in as HTTP. Therefore, the Web Console component should allow HTTP connections. The below additional setting is required be added to accept HTTP incoming connections to the Web Console.

Note: consider configuring this additional setting on a Client Computer Group level for easier implementation



Name forceHttps

Category WebConsole

Type BOOLEAN

Value false

Enable ☑

Step 3 - Adapt the server.xml file on the Web Console servers

By default, Tomcat in this configuration does not allow connections from a reverse proxy; therefore, the connection fails. This connection failure is caused by various security features that rely on HTTPS connections and which we are offloading to the network load balancer.

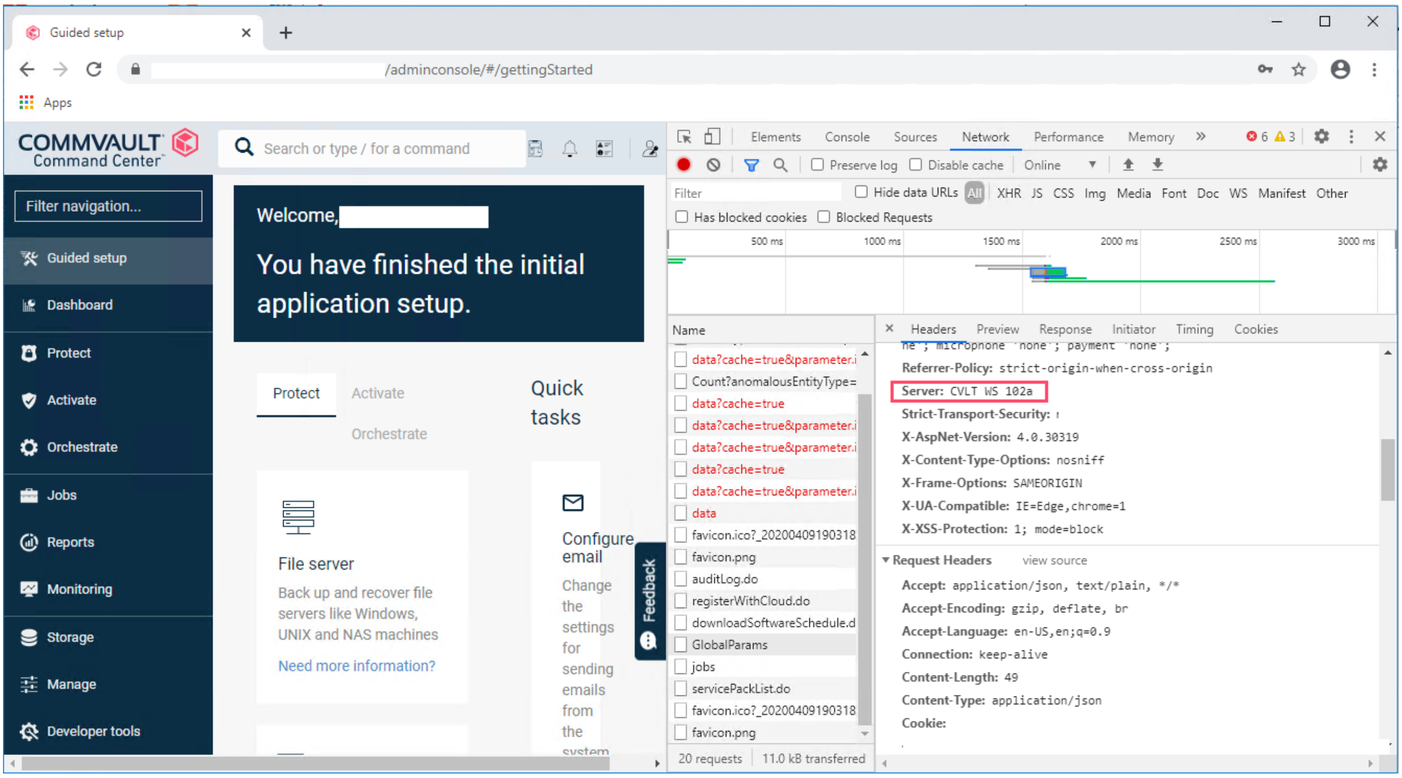
To allow the Tomcat component to accept connections from a reverse proxy, the server configuration XML requires a few modifications. The file can be found in the following location: "<install_directory>\Apache\Conf\server.xml."

Modify the connector port configuration with the following guidance:

- Modify the "Server" attribute to reflect the web console server used during the user session. This is an optional configuration and is not required to allow the architecture to work.
- The attributes "scheme, secure, proxyPort & ProxyName" are used to force Tomcat to reply in HTTPS on any HTTP requests. This configuration is mandatory to allow the SAML authentication to work. Without this specific modification, the SAML request generated by Tomcat is prefixed with "HTTP" and revoked by the Identity Provider.

```
<Connector port="80" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="443" server="CVWC1" compression="on" noCompressionUserAgents="gozilla,traviata" compressionMinSize="500" compressableMimeType="text/html,text/json,application/json,text/xml,text/plain,application/javascript,text/css,text/javascript,text/js" useSendfile="false" scheme="https" secure="true" proxyPort="443" proxyName="the.loadbalancer.fqdn" />
```

The "Server" attribute can be accessed using the developer tools of your browser as highlighted in the screenshot.



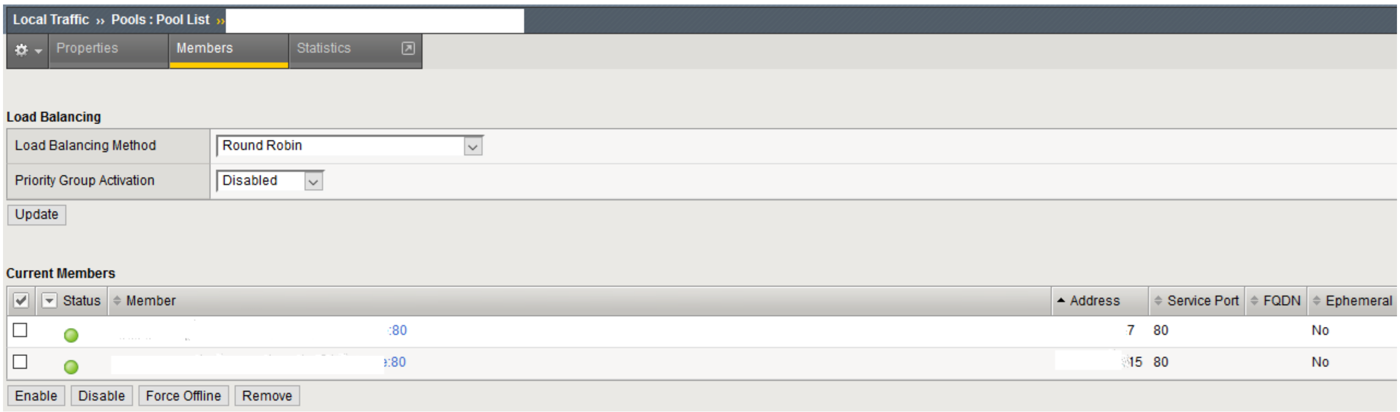
```
ne'; microphone 'none'; payment 'none';
Referrer-Policy: strict-origin-when-cross-origin
Server: CVLT WS 102a
Strict-Transport-Security: ı
X-AspNet-Version: 4.0.30319
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-UA-Compatible: IE=Edge,chrome=1
X-XSS-Protection: 1; mode=block
▼ Request Headers    view source
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Content-Length: 49
Content-Type: application/json
Cookie:
```

Step 4 - Configure the load balancer

All the required Commvault configuration has been performed throughout steps 1 to 3. This step highlights the configuration required on the Network Load Balancer. The below information below has been collected on an F5 Network Load Balancer in conjunction with the customer's network administrator.

Create a new Pool and:

- Configure a Virtual IP Address to be used during end-user access.
- Accept HTTP (TCP/80) and HTTPS (TCP/443) traffic.
- Configure the load balancer to redirect any HTTP requests on the front-end automatically to HTTPS.
- Configure the Load Balancing Method in "*active/active*" setup using "*Round-Robin*."



Local Traffic » Pools : Pool List » [redacted]

Load Balancing

Load Balancing Method	Round Robin
Priority Group Activation	Disabled

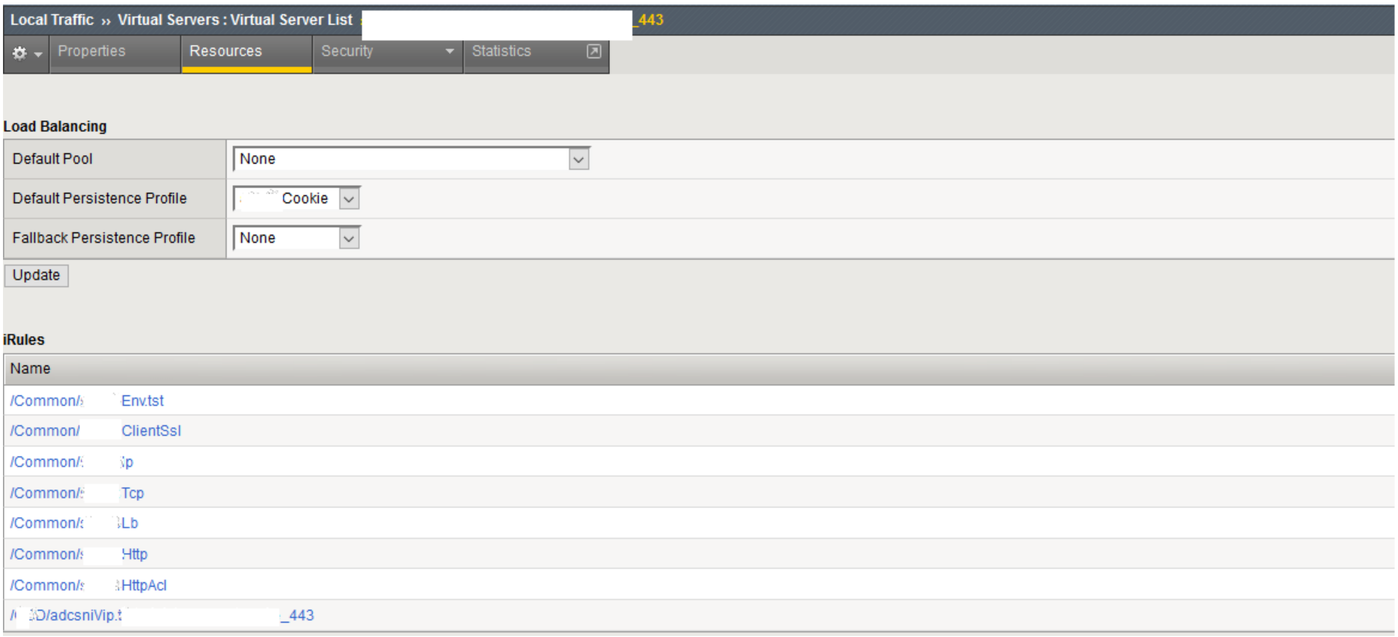
Update

Current Members

☑	Status	Member	Address	Service Port	FQDN	Ephemeral
□	●	[redacted]:80	[redacted]7	80		No
□	●	[redacted]:80	[redacted]15	80		No

Enable | Disable | Force Offline | Remove

- Configure sticky sessions using an inserted cookie in the session.



Local Traffic » Virtual Servers : Virtual Server List [redacted]_443

Load Balancing

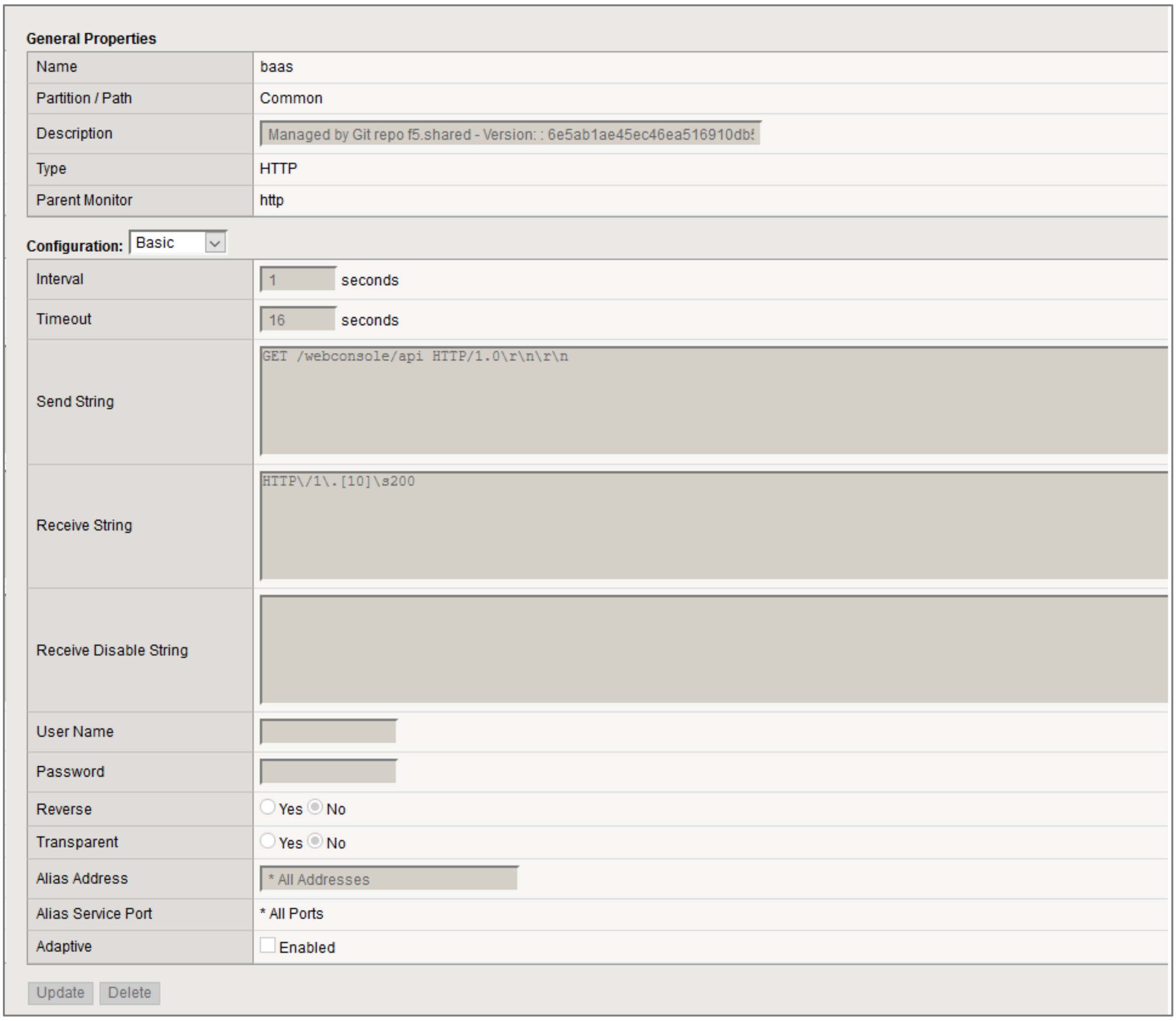
Default Pool	None
Default Persistence Profile	[redacted]Cookie
Fallback Persistence Profile	None

Update

iRules

Name
/Common/[redacted]Env.tst
/Common/[redacted]ClientSsl
/Common/[redacted]p
/Common/[redacted]Tcp
/Common/[redacted]Lb
/Common/[redacted]Http
/Common/[redacted]HttpAcl
/[redacted]D/adcsniVip.[redacted]_443

- Configure the health monitors by both using "ICMP" and an "HTTP error code check." In the example below, you will notice a recurrent check on the page "/webconsole/api" on each webconsole to verify if it responds with HTTP code 200 "OK."



General Properties

Name	baas
Partition / Path	Common
Description	Managed by Git repo f5.shared - Version: : 6e5ab1ae45ec46ea516910db
Type	HTTP
Parent Monitor	http

Configuration: Basic

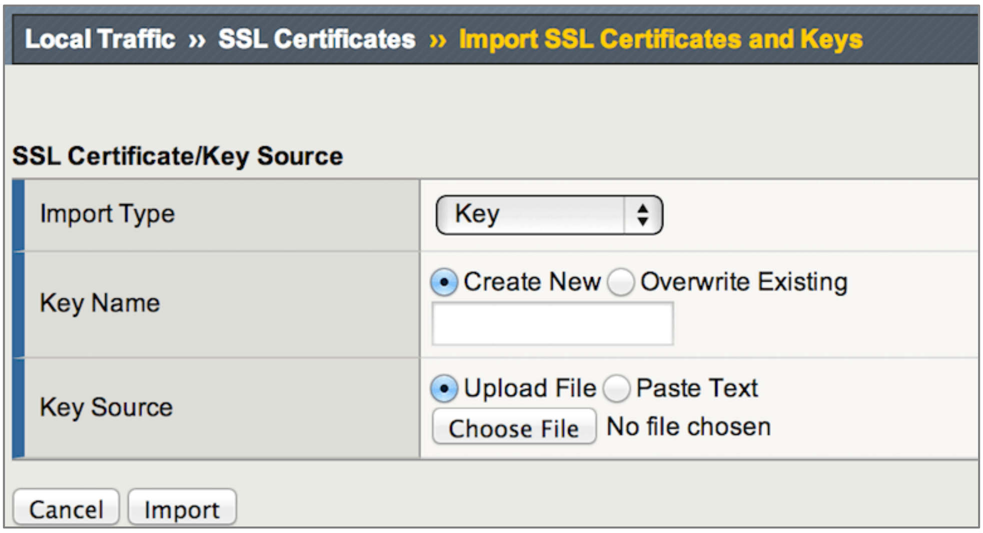
Interval	1 seconds
Timeout	16 seconds
Send String	`GET /webconsole/api HTTP/1.0\r\n\r\n`
Receive String	`HTTP\/1\.[10]\s200`
Receive Disable String	
User Name	
Password	
Reverse	○ Yes ● No
Transparent	○ Yes ● No
Alias Address	* All Addresses
Alias Service Port	* All Ports
Adaptive	□ Enabled

- To avoid the following error: "StatelessCookieFilter:isValidReferer:280 - Referer check failed. Referer: *https://myserver.com/webconsole/login/index.jsp?disableSSO* RequestUrl: *http://myserver.com:80*" in the logs. We modified the "Connector" tag for "HTTP" in the "server.xml" with the attributes "scheme, secure, proxyName & proxyPort." Therefore, the NLB can be configured with a default iRule.

```
when HTTP_REQUEST priority 500 {
SSL::disable serverside
# Check which host the client requested


        switch -glob -- $httpHost \

                "<webconsole_fqdn>" {
                pool <pool_name>

        } \


        default {

                # Block all other requests
                HTTP::respond 403 content "Sorry, you don't have access to the web site"
                call ****Procs::accesslog_v1_403 "$ipClient" "$httpHost" "$httpRequestTime" "$httpMethod"
                "$httpUri" "$httpVersion" "$httpReferer" "$httpUserAgent" "$sslClientCipherNameLog"
                "$sslClientCipherVersionLog" "$sslClientCipherBitsLog" "$vsName" "$custEnv" "$partitionName"

        }


}
```

Step 5 – Configure an SSL certificate for the virtual site

Configure an SSL certificate on the network load balancer to secure the virtual site with HTTPS.



Local Traffic » SSL Certificates » Import SSL Certificates and Keys

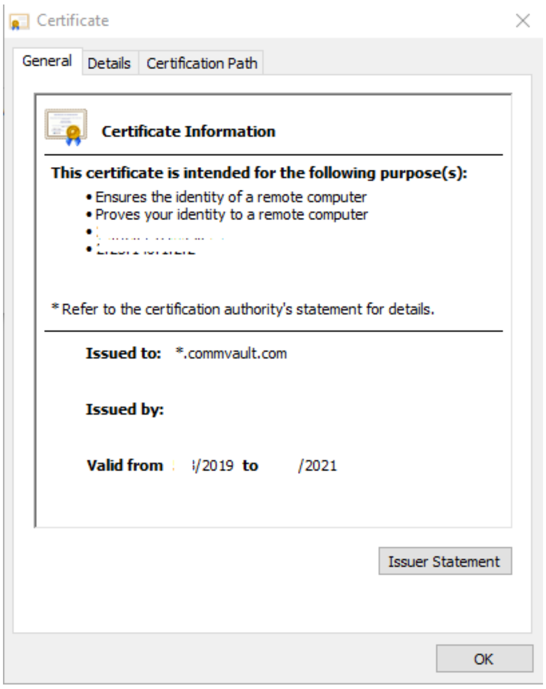
SSL Certificate/Key Source

Import Type	Key
Key Name	● Create New ○ Overwrite Existing
Key Source	● Upload File ○ Paste Text Choose File No file chosen

Step 6 – Test your configuration

Open a web browser and connect to http://loadbalancedwebconsole.domain.fqdn and verify:

- If the web page successfully opens.
- If the network load balancer performed an HTTP to HTTPS redirection.
- The web page uses a valid certificate.



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer

* Refer to the certification authority's statement for details.

Issued to: *.commvault.com

Issued by:

Valid from /2019 to /2021

FAQs

Below are the most frequently asked questions from customers.

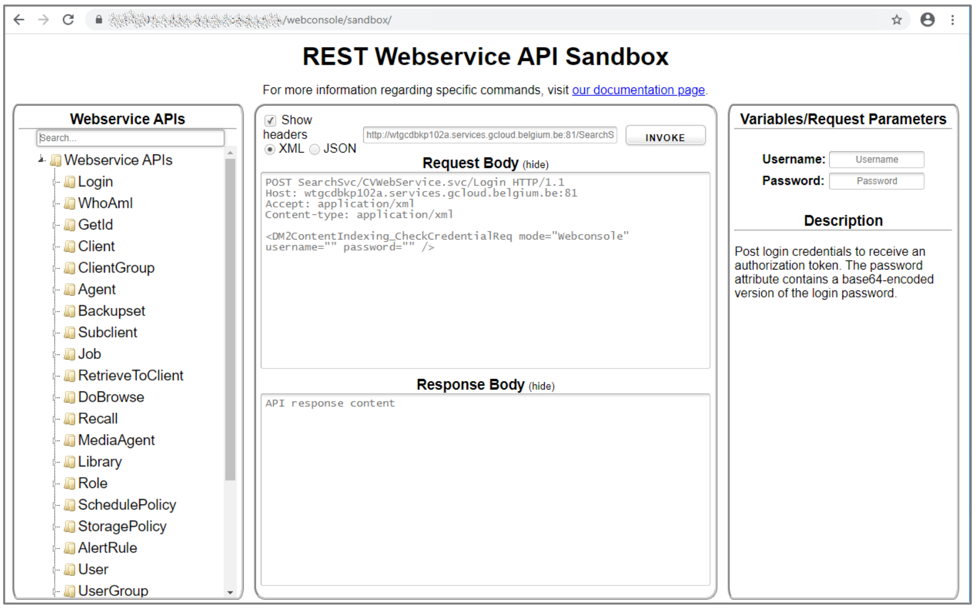
Does Commvault support load-balanced web console servers?

Yes, we do.

Why would you put a Network Load Balancer (NLB) in front of the Web Console servers?

There are several reasons to implement this topology:

- **Security**: The NLB functions as a reverse proxy. This means the NLB is exposed to the outside world, whereby the Web Console server is residing in a secured DMZ network. Therefore, there is no reason to decouple the Web Console role from the Web Server role on two different systems.
- **Port forwarding**: On the NLB, you can easily configure TCP port forwarding. For example, every time a user connects on HTTP, the connection is automatically forwarded to HTTPS.
- **Availability**: When configured correctly, the NLB load balances the sessions over all stacks in this configuration. If one of the stacks goes offline, the NLB will not relay any new connections to it.
- **DDoS prevention**: DDoS or Distributed Denial of Service attacks are purposed to bring an online service or website unavailable by flooding it with unwanted traffic from several computers. An NLB has the built-in intelligence to cope with these attacks and ensure your business applications remain unaffected.
- **Resource-optimization**: By removing the need for encrypting and decrypting HTTPS traffic with the use of SSL offloading. The back end of the NLB can be correctly configured using HTTP.
- **Ease-of-management**: Customers like to secure the Web Console with a certificate signed by a Certificate Authority (CA). As certificates have expiration dates, the renewal process can become frequent. Changing the certificate on an NLB is straightforward using a wizard and can be completed in less than 10 minutes.
- **Security**: An NLB can be used to fence off certain virtual sites on the Web Console server. For example, perform an automatic redirect to "/adminconsole" and disallow users to connect to anything else. This approach allows you to fence of the webconsole API Sandbox Virtual Site.



[redacted]/webconsole/sandbox/

REST Webservice API Sandbox

For more information regarding specific commands, visit our documentation page.

☑ Show headers

● XML ○ JSON

http://wtgcdbkp102a.services.gcloud.belgium.be:81/SearchS

Request Body (hide)

```
POST SearchSvc/CVWebService.svc/Login HTTP/1.1
Host: wtgcdbkp102a.services.gcloud.belgium.be:81
Accept: application/xml
Content-type: application/xml

<DM2ContentIndexing_CheckCredentialReq mode="Webconsole"
username="" password="" />
```

Response Body (hide)

```
API response content
```

Variables/Request Parameters

Username: Username

Password: Password

Description

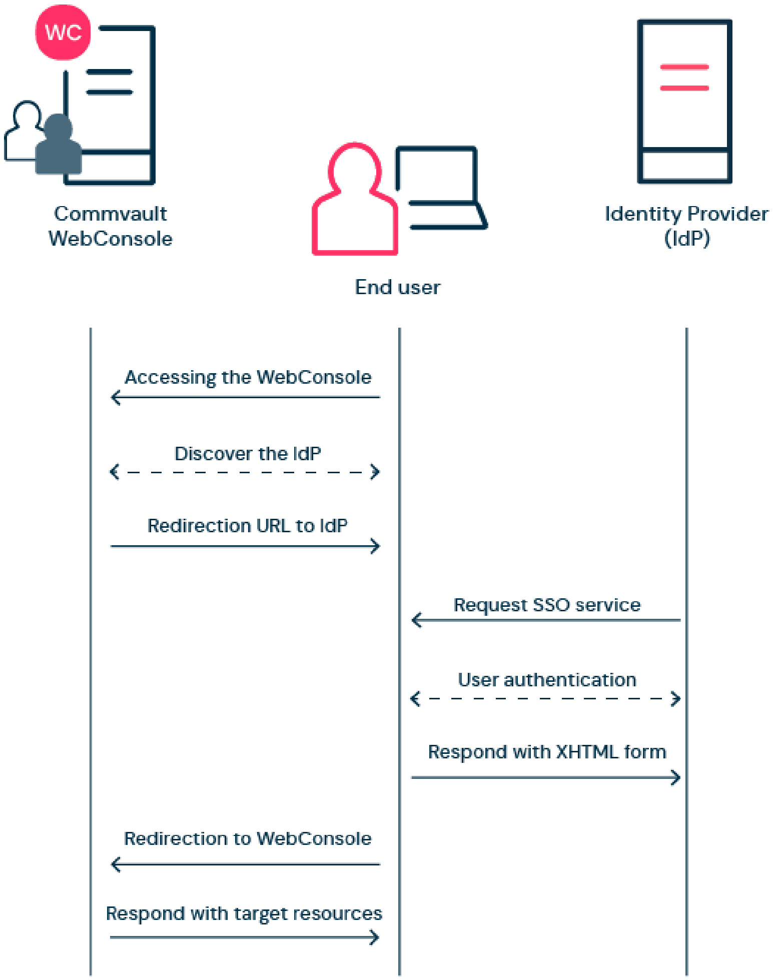
Post login credentials to receive an authorization token. The password attribute contains a base64-encoded version of the login password.

What is SSL offloading?

SSL offloading relieves the Web Console servers of the processing burden of encrypting and/or decrypting traffic sent via SSL.

Does SSL offloading impact authentication mechanisms such as SAML?

As SAML is performed through the web browser on the client system (see diagram below), the entire process is transparent for the identity provider.



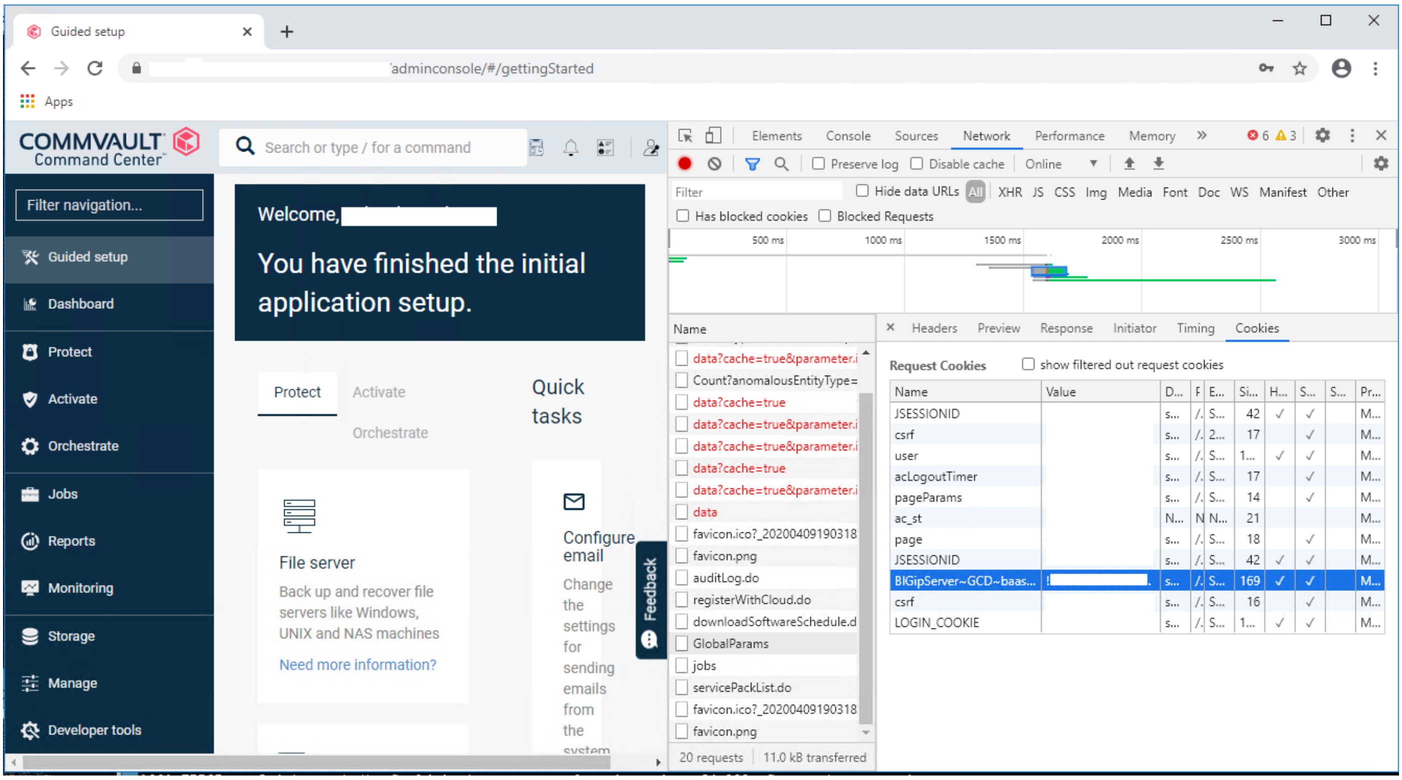
What are sticky sessions?

Sticky sessions or session-affinity is the technique to route requests for a session to the same machine that serviced the first request for that session. This is performed to ensure that any in-proc session data is not lost.

Since requests for a single user session is always routed to the same Web Console server, sticky sessions can cause an uneven load distribution across servers. The configuration of sticky sessions is mandatory!

Network Load Balancers usually provide two different techniques to ensure all requests for a user is relayed to the same Web Console Server.

- **Cookie-based**: The NLB adds a cookie in the traffic. Each time the browser sends a new request, the cookie is used by the NLB to relay the request to the same Web Console server in the background. This approach is preferred.
- **IP-based**: The NLB keeps track of all connections based on the source IP address. Each time a new request is triggered, the NLB verifies the IP and relays the connection to the documented Web Console server in the connection history. This approach has a lower preference.



Welcome, [redacted]

You have finished the initial application setup.

Protect | Activate | Orchestrate

Quick tasks

File server

Back up and recover file servers like Windows, UNIX and NAS machines

Need more information?

Configure email

Change the settings for sending emails from the system

Name
- data?cache=true&parameter.i
- Count?anomalousEntityType=
- data?cache=true
- data?cache=true&parameter.i
- data?cache=true&parameter.i
- data?cache=true
- data?cache=true&parameter.i
- data
- favicon.ico?_20200409190318
- favicon.png
- auditLog.do
- registerWithCloud.do
- downloadSoftwareSchedule.d
- GlobalParams
- jobs
- servicePackList.do
- favicon.ico?_20200409190318
- favicon.png

20 requests | 11.0 kB transferred

Request Cookies ☐ show filtered out request cookies

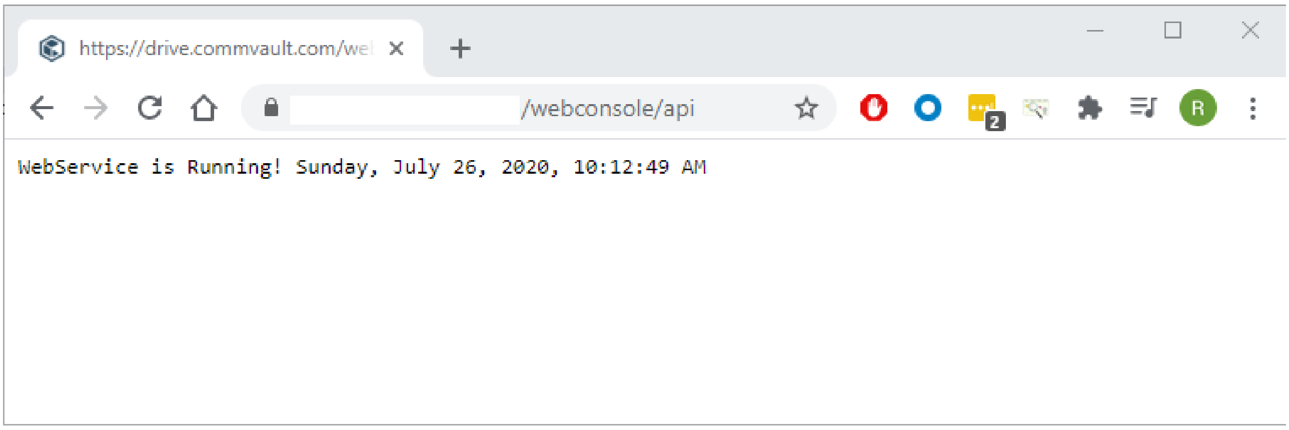
Name	Value	D...	P	E...	Si...	H...	S...	S...	Pr...
JSESSIONID		s...	/.	S...	42	✓	✓		M...
csrf		s...	/.	2...	17		✓		M...
user		s...	/.	S...	1...	✓	✓		M...
acLogoutTimer		s...	/.	S...	17		✓		M...
pageParams		s...	/.	S...	14		✓		M...
ac_st		N...	N	N...	21				M...
page		s...	/.	S...	18		✓		M...
JSESSIONID		s...	/.	S...	42	✓	✓		M...
BIGipServer~GCD~baas...	![redacted].	s...	/.	S...	169	✓	✓		M...
csrf		s...	/.	S...	16		✓		M...
LOGIN_COOKIE		s...	/.	S...	1...	✓	✓		M...

What's the best way to check if a "Web Console / Web Server"-stack is operational?

As the stack is relying on both the Web Console - and Web Server component, it's mandatory to perform and end-to-end health check from the Network Load Balancer. This is not feasible using:

- An ICMP check (ping reply) as it verifies the system is up, it does not check the services.
- An TCP port availability check as this will only verify if the Web Console service is running but does not determine the availability of the Web Server component in the background.

By polling the URL "<webconsole_servername>/webconsole/api" and verifying the HTTP return code, we can determine the health state of the entire stack. If the code is something else than "HTTP 200 OK", the stack is impacted and should not be used anymore. (List of HTTP status codes - Wikipedia)



```
WebService is Running! Sunday, July 26, 2020, 10:12:49 AM
```

Summary

With the Commvault Command Center, webserver, and Web Console becoming increasingly more crucial during daily data management administration tasks, implementing a load-balanced Web Console server allows you to achieve a new level of resilience. This resiliency helps you to meet predefined service level agreements, optimize resources, and increase the security of the stack.

The functionalities and benefits provided by the NLB include:

- Web Console servers reside behind a reverse proxy. This topology avoids any direct communication from the "public network/internet" towards the web console server(s). – Effectively increasing the overall security.
- An NLB listens on a predefined port (e.g. HTTPS) or predefined set of ports (e.g. HTTP & HTTPS). The NLB can be configured to perform port forwarding to ensure all network traffic is secured with HTTPS (HTTP > HTTPS). Additionally, the NLB can be used to fence certain virtual sites.
- A network load balancer is designed to cope with DDoS attacks. In case of a DDoS attack, the service is left unaffected, and the administrators/customers can still perform their daily management tasks.
- A network load balancer simplifies the rotation of certificates. Certificates can be updated in a few clicks.
- A network load balancer continuously performs a health check of the web console servers in the background. In case one of the peers becomes unavailable, the NLB will not relay any new connections to it.