Discovery 6: Configure High Availability

Introduction

In this activity, you will learn how to implement high availability for NSO. You will work with two NSO-in-Docker containers, *nso-primary* and *nso-secondary.* They are already prepared with the system installation. You will configure high availability so that in case one of the nodes goes down, the provisioning process can still go on the remaining node.

After completing this activity, you will be able to:

- Configure NSO built-in high availability.
- Test and remediate different failure scenarios.
- Configure virtual IP (VIP) for northbound systems to seamlessly switch to the working NSO node if there is a failure using the *tailf-hcc* package.

Job Aid

The following job aid is available to help you complete the lab activities:

- This Lab Guide

The following table contains passwords that you might need.

Device	Username	Password
student-VM	student	1234QWer
nso-server	student	1234QWer

Required Resources

The following resources and equipment are required for completing the activities in this lab guide:

- PC or laptop with a web browser
- Access to the internet

Command List

The following are the most common commands that you will need:

Linux Shell:

Command	Comment
source /opt/ncs/ncs-6.1/ncsrc	Source NSO environmental variable in Docker container.
ls|ll	Display contents of the current directory.

Command	Comment
cd	Move directly to user home directory.
cd ..	Exit out of current directory.
cd test	Move into the "test" folder, which is a subfolder of the current directory.
cd /home/student	Move into the "nso300" folder by specifying the direct path to it starting from the root of the directory system.
ncs_cli -C	Log in to NSO CLI directly from local server.

NSO CLI:

Command	Comment
switch cli	Change CLI style.
show ?	Display all command options for current mode.
configure	Enter configuration mode.
commit	Commit new configuration (configuration mode only command).
show configuration	Display new configuration that has not yet been committed (configuration mode only command).

Makefile commands for Docker environment:

Command	Comment
make build	Builds the main NSO Docker image.
make testenv-start	Starts the NSO Docker environment.
make testenv-stop	Stops the NSO Docker environment.
make testenv-build	Recompiles and reloads the NSO packages.
make testenv-cli	Enters the NSO CLI of the NSO Docker container.
make testenv-shell	Enters the Linux shell of the NSO Docker container.
make dev-shell	Enters the Linux shell of the NSO Docker development container.

Command Syntax Reference

This lab guide uses the following conventions for command syntax:

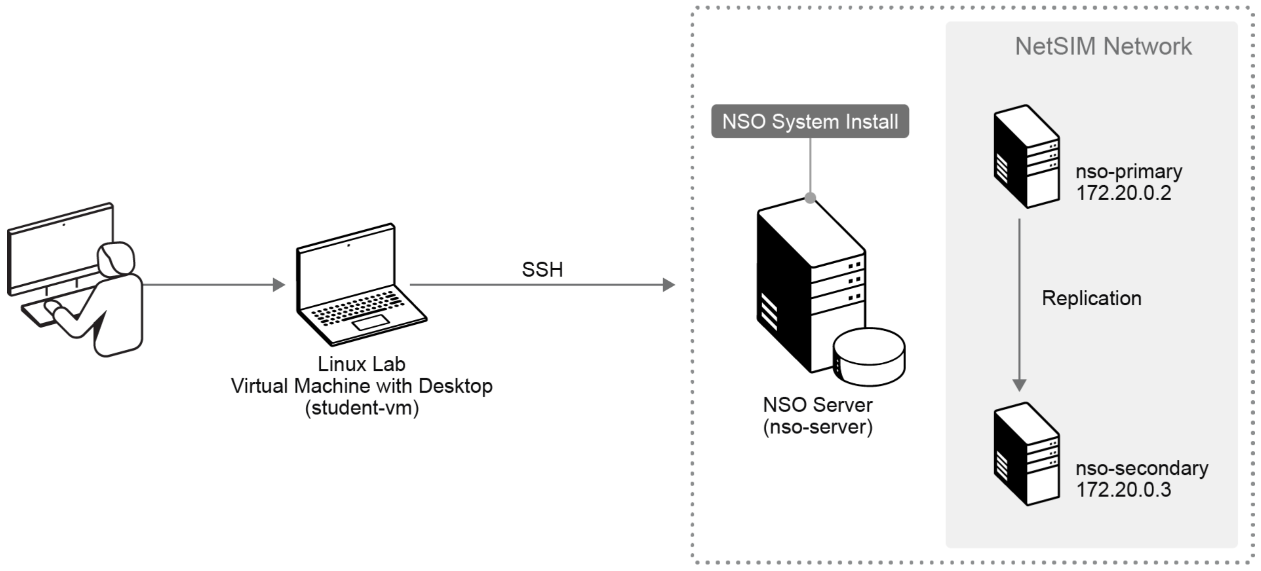
Formatting	Description and Examples
show running config	Commands in steps use this formatting.
Example	Type **show running config**
Example	Use the **name** command.

Formatting	Description and Examples
`show running config`	Commands in CLI outputs and configurations use this formatting.
highlight	CLI output that is important is highlighted.
Example	`student@student-vm:~$ ncs -version` `6.1`
Example	Save your current configuration as the default **startup config**. `Router Name# copy running startup`
brackets ([ ])	Indicates optional element. You can choose one of the options.
Example:	`(config-if)# frame-relay lmi-type {ansi|cisco|q933a}`
italics font	Arguments for which you supply values.
Example	Open file **ip tcp window-size** *bytes*
angle brackets (<>)	In contexts that do not allow italics, arguments for which you supply values are enclosed in angle brackets [<>]. Do not type the brackets when entering the command.
Example	If the command syntax is **ping** <*ip_address*>, you enter ping *10.0.0.102*
string	A non-quoted set of characters. Type the characters as-is.
Example	(config)# **hostname MyRouter**
vertical line (|)	Indicates that you enter one of the choices. The vertical line separates choices. Do not type the vertical line when entering the command.
Example	If the command syntax is **show ip route|arp**, you enter either **show ip route** or **show ip arp**, but not both.

Lab Topology Information

Your lab session is your own personal sandbox. Whatever you do in your session will not be reflected in anyone else's session. There are two topologies. Your lab environment is a Linux server (Student-VM) acting as a jumphost, and a Linux server (NSO-server) acting as an Docker environment that consists of two NSO Docker containers with your NSO installation.

Topology



Task 1: Set Up High Availability

In this task, you will configure two NSO nodes for high availability. The two nodes (*nso-primary* and *nso-secondary*) are NSO-in-Docker containers. You can treat the nodes as two separate virtual machines that are running in your lab instance.

Activity

Step 1

Connect to the Student-VM.

You can connect to the server either by choosing the **Student-VM** from the device list or by clicking on the **Student-VM** icon in the topology map.

Step 2

Open the terminal window.

Open the terminal window by clicking the **Terminal** icon in the bottom bar.

```
student@student-vm:~$
```

Step 3

Connect to the **nso-server** NSO server.

Connect to the **nso-server** NSO server with the student user using the SSH client. The authentication is already preconfigured with public key authentication, therefore the password is not needed. The prompt will change, stating that you are now connected to the nso-server.

```
student@student-vm:~$ ssh student@nso-server
Last login: Tue Oct  3 09:14:42 2023 from 10.0.0.102
student@nso-server:~$
```

Step 4

Enter the *nso-ha* directory.

Use the **cd nso-ha** command.

```
student@nso-server:~$ cd nso-ha
student@nso-server:~/nso-ha$
```

Step 5

List the Docker containers that are currently running on your virtual machine by using the **docker ps -a** command.

Locate the primary and secondary NSO containers.

```
student@nso-server:~/nso-ha$ docker ps -a
CONTAINER ID   IMAGE                                              COMMAND
CREATED               STATUS                           PORTS
NAMES
f0315c8e9f0b   nso303.gitlab.local/nso-ha/nso:6.1-student   "/run-
nso.sh"       About a minute ago   Up About a minute (healthy)   22/
tcp, 80/tcp, 443/tcp, 830/tcp, 4334/tcp
testenv-nso-ha-6.1-student-nso-secondary
2c21fe68dc17   nso303.gitlab.local/nso-ha/nso:6.1-student   "/run-
nso.sh"       About a minute ago   Up About a minute (healthy)   22/
tcp, 80/tcp, 443/tcp, 830/tcp, 4334/tcp
testenv-nso-ha-6.1-student-nso-primary
5b243c1b17f3   gitlab/gitlab-ee:13.4.0-ee                   "/assets/
wrapper"   3 months ago         Up 5 hours (healthy)
0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/
tcp, 0.0.0.0:8022->22/tcp, :::8022->22/tcp   gitlab_web_1
student@nso-server:~/nso-ha$
```

Step 6

Connect to the primary NSO Linux shell.

Use the **make testenv-shell NSO=<nso>** command.

```
student@nso-server:~/nso-ha$ make testenv-shell NSO=primary
docker exec -it testenv-nso-ha-6.1-student-nso-primary bash -l
root@2c21fe68dc17:/#
```

Step 7

Open the *ncs.conf* configuration file in the **/etc/ncs** directory.

The ncs configuration in this lab is already set to work with a high availability environment.

```
root@2c21fe68dc17:/# vim /etc/ncs/ncs.conf
```

Step 8

Make sure that the northbound NETCONF communication is enabled.

The configuration is located in the <netconf-north-bound> section.

```
...
  <netconf-north-bound>
    <enabled>true</enabled>
    <transport>
      <ssh>
        <enabled>true</enabled>
        <ip>0.0.0.0</ip>
        <port>830</port>
        <extra-listen>
          <ip>::</ip>
          <port>830</port>
        </extra-listen>
      </ssh>
      <tcp>
        <enabled>false</enabled>
        <ip>127.0.0.1</ip>
        <port>2023</port>
      </tcp>
    </transport>
  </netconf-north-bound>
...
```

Step 9

Make sure that high availability is enabled.

The configuration is located in the <ha> section.

```
...
  <ha>
    <enabled>true</enabled>
  </ha>
...
```

Step 10

Make sure that the WebUI is enabled for VIP failover testing.

The configuration is located in the <webui> section.

```
...
  <webui>
    <enabled>true</enabled>
```

```
      <transport>
        <tcp>
          <enabled>true</enabled>
          <ip>0.0.0.0</ip>
          <port>80</port>
          <extra-listen>
            <ip>::</ip>
            <port>80</port>
          </extra-listen>
        </tcp>
        <ssl>
          <enabled>false</enabled>
          <ip>0.0.0.0</ip>
          <port>443</port>
          <key-file>${NCS_CONFIG_DIR}/ssl/cert/host.key</key-file>
          <cert-file>${NCS_CONFIG_DIR}/ssl/cert/host.cert</cert-file>
          <extra-listen>
            <ip>::</ip>
            <port>443</port>
          </extra-listen>
        </ssl>
      </transport>
      <cgi>
        <enabled>true</enabled>
        <php>
          <enabled>false</enabled>
        </php>
      </cgi>
    </webui>
  ...
```

Step 11

Make sure that the CLI prompt is correct for the primary node.

High availability prompts are formatted by adding "-primary" to every CLI prompt for easier orientation.

```
  ...
    <cli>
      <enabled>true</enabled>
      <!-- Use the builtin SSH server -->
      <ssh>
        <enabled>true</enabled>
        <ip>0.0.0.0</ip>
        <port>22</port>
        <extra-listen>
          <ip>::</ip>
          <port>22</port>
        </extra-listen>
      </ssh>
      <prompt1>\u@ncs-primary&gt; </prompt1>
      <prompt2>\u@ncs-primary% </prompt2>
      <c-prompt1>\u@ncs-primary# </c-prompt1>
      <c-prompt2>\u@ncs-primary(\m)# </c-prompt2>
      <restricted-file-access>true</restricted-file-access>
      <show-commit-progress>true</show-commit-progress>
    <suppress-commit-message-context>maapi</suppress-commit-message-
context>
    <suppress-commit-message-context>system</suppress-commit-message-
context>
  </cli>
...
```

Step 12

Exit the file editor.

Exit the file editor by using the **:q** command and pressing the **Enter** key on your keyboard.

Step 13

Exit the container shell.

Use the **exit** command.

Step 14

Repeat steps 7 to 10 and verify the ncs configuration for the secondary NSO node.

Use the **make testenv-cli NSO=secondary** command to enter the container shell.

Step 15

Change the prompts on the secondary node to include **secondary** instead of **primary** in the name.

To enter the edit mode in vim, press the **Insert** key.

```

...
  <cli>
    <enabled>true</enabled>
    <!-- Use the builtin SSH server -->
    <ssh>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>22</port>
      <extra-listen>
        <ip>::</ip>
        <port>22</port>
      </extra-listen>
    </ssh>
    <prompt1>\u@ncs-secondary&gt; </prompt1>
    <prompt2>\u@ncs-secondary% </prompt2>
    <c-prompt1>\u@ncs-secondary# </c-prompt1>
    <c-prompt2>\u@ncs-secondary(\m)# </c-prompt2>
    <restricted-file-access>true</restricted-file-access>
    <show-log-directory>${NCS_LOG_DIR}</show-log-directory>
    <show-commit-progress>true</show-commit-progress>
    <suppress-commit-message-context>maapi</suppress-commit-message-
context>
    <suppress-commit-message-context>system</suppress-commit-message-
context>
  </cli>
```

```
...
```

Step 16

Save the file and exit the file editor.

To exit the edit mode, press the **Esc** key, and then use the **:wq** command and pressing the **Enter** key on your keyboard to save the file and exit the file editor.

Step 17

Reload the ncs configuration and exit the container shell.

Use the the **ncs --reload** command and exit the container shell.

```
root@f2a232de68f2:/# ncs --reload
root@f2a232de68f2:/# exit
student@nso-server:~/nso-ha$
```

Step 18

Copy the **tailf-hcc** package from the **~/packages** to the **packages** folder.

You will need the *tailf-hcc* package to enable VIP functionality.

```
student@nso-server:~/nso-ha$ cp -r ../packages/tailf-hcc packages/
student@nso-server:~/nso-ha$
```

Step 19

Rebuild and reload the packages.

By using the **make testenv-build** command, packages in both the primary and secondary nodes will be rebuilt.

Make sure that the **tailf-hcc** package is reloaded successfully.

```
student@nso-server:~/nso-ha$ make testenv-build
for NSO in $(docker ps --format '{{.Names}}' --filter label=testenv-
nso-ha-6.1-student --filter label=nidtype=nso); do \
        echo "-- Rebuilding for NSO: ${NSO}"; \
        docker run -it --rm -v /home/student/nso-ha:/src --volumes-from
${NSO} --network=container:${NSO} -e NSO=${NSO} -e PACKAGE_RELOAD= -e
SKIP_LINT= -e PKG_FILE=nso303.gitlab.local/nso-ha/package:6.1-student
nso303.gitlab.local/nso-ha/cisco-nso-dev:6.1 /src/nid/testenv-build; \
done
-- Rebuilding for NSO: testenv-nso-ha-6.1-student-nso-secondary
...
...

>>> System upgrade is starting.
>>> Sessions in configure mode must exit to operational mode.
>>> No configuration changes can be performed until upgrade has
```

```
completed.
>>> System upgrade has completed successfully.
reload-result {
    package tailf-hcc
    result true
}
-- Rebuilding for NSO: testenv-nso-ha-6.1-student-nso-primary
(package-meta-data.xml|\.cli1|\.yang1)
make: Entering directory '/var/opt/ncs/packages/tailf-hcc/src'
...
...

>>> System upgrade is starting.
>>> Sessions in configure mode must exit to operational mode.
>>> No configuration changes can be performed until upgrade has
completed.
>>> System upgrade has completed successfully.
reload-result {
    package tailf-hcc
    result true
}
student@nso-server:~/nso-ha$
```

Step 20

Connect to the NSO CLI of the primary node.

To enter the NSO CLI of the primary node, use the **make testenv-cli NSO=primary** command.

```
student@nso-server:~/nso-ha$ make testenv-cli NSO=primary
docker exec -it testenv-nso-ha-6.1-student-nso-primary bash -lc
'ncs_cli -Cu admin'

User admin last logged in 2024-02-08T20:08:37.788814+00:00, to
2c21fe68dc17, from 127.0.0.1 using cli-console
admin connected from 127.0.0.1 using console on 2c21fe68dc17
admin@ncs-primary#
```

Step 21

Enter the config mode and configure high availability settings.

Use the following commands:

```
@ncs- primary # config
Entering configuration mode terminal
admin@ncs-primary(config)# high-availability token NSO303
admin@ncs-primary(config)# high-availability settings enable-failover
true
admin@ncs-primary(config)# high-availability settings start-up assume-
nominal-role true
admin@ncs-primary(config)# high-availability settings start-up join-ha
true
admin@ncs-primary(config)# high-availability settings reconnect-
```

```
interval 2
admin@ncs-primary(config)# high-availability ha-node nso-primary
address 172.21.0.2 nominal-role primary
admin@ncs-primary(config-ha-node-nso-primary)# exit
admin@ncs-primary(config)# high-availability ha-node nso-secondary
address 172.21.0.3 nominal-role secondary failover-primary true
admin@ncs-primary(config-ha-node-nso-secondary)# top
admin@ncs-primary(config)# commit
Commit complete.
admin@ncs-primary(config)# high-availability enable

result enabled

admin@ncs-primary(config)#
```

Step 22

Exit the config mode and the NSO CLI.

Use the **exit** command.

```
admin@ncs-primary(config)# exit
admin@ncs-primary# exit
student@nso-server:~/nso-ha$
```

Step 23

Enter the NSO CLI of the secondary node and enter the config mode.

To enter the NSO CLI of the secondary node, use the **make testenv-cli NSO=secondary** command.

```
student@nso-server:~/nso-ha$ make testenv-cli NSO=secondary
docker exec -it testenv-nso-ha-6.1-student-nso-secondary bash -lc
'ncs_cli -Cu admin'

User admin last logged in 2024-02-08T20:05:49.767061+00:00, to
f0315c8e9f0b, from 127.0.0.1 using cli-console
admin connected from 127.0.0.1 using console on f0315c8e9f0b
admin@ncs-secondary# config
Entering configuration mode terminal
admin@ncs-secondary(config)#
```

Step 24

Configure high availability settings for the secondary node only with minimum required parameters to enable high-availability, commit the changes, and exit the configuration mode.

The rest of the configuration will be replicated when the connection between nodes is established. Exit the configuration mode after the commit.

```
admin@ncs-secondary(config)# high-availability token NSO303
```

```
admin@ncs-secondary(config)# high-availability ha-node nso-primary
address 172.21.0.2 nominal-role primary
admin@ncs-secondary(config-ha-node-nso-primary)# exit
admin@ncs-secondary(config)# high-availability ha-node nso-secondary
address 172.21.0.3 nominal-role secondary
admin@ncs-secondary(config-ha-node-nso-secondary)# commit
Commit complete.
admin@ncs-secondary(config-ha-node-nso-secondary)# top
admin@ncs-secondary(config)# exit
```

Step 25

Enable the high availability configuration with the **high-availability enable** command and set the node to be secondary node to connect to the primary.

To successfully connect for the first time and retrieve the rest of the high-availability configuration, the **be-secondary-to node** action needs to be invoked.

```
admin@ncs-secondary# high-availability enable
result enabled
admin@ncs-secondary# high-availability be-secondary-to node nso-primary
result Attempting to be secondary to node nso-primary
admin@ncs-secondary# show high-availability
high-availability enabled
high-availability status mode secondary
high-availability status current-id nso-secondary
high-availability status assigned-role secondary
high-availability status be-secondary-result initialized
high-availability status primary-id nso-primary
high-availability status read-only-mode false
admin@ncs-secondary# exit
student@student-vm:~/nso-ha$
```

Step 26

Verify if the missing configuration has been replicated. Exit the NSO CLI.

Since the token is encrypted, it might be different from the output below. The rest of the high-availability configuration should now be present on the nso-secondary node as well.

```
admin@ncs-secondary# show running-config high-availability
high-availability token $9$GcKRhK/WZkRePMeX2pS3irLWGwI6jvQL+lgQ9McB//E=
high-availability ha-node nso-primary
 address      172.21.0.2
 nominal-role primary
!
high-availability ha-node nso-secondary
 address          172.21.0.3
 nominal-role     secondary
 failover-primary true
!
high-availability settings enable-failover true
high-availability settings start-up assume-nominal-role true
```

```
high-availability settings start-up join-ha true
high-availability settings reconnect-interval 2
admin@ncs-secondary# exit
student@nso-server:~/nso-ha$
```

Step 27

Enter the NSO CLI of the primary node.

To enter the NSO CLI of the primary node, use the **make testenv-cli NSO=primary** command.

```
student@nso-server:~/nso-ha$ make testenv-cli NSO=primary
docker exec -it testenv-nso-ha-6.1-student-nso-primary bash -lc
'ncs_cli -Cu admin'

User admin last logged in 2024-02-08T20:29:35.971135+00:00, to
2c21fe68dc17, from 127.0.0.1 using cli-console
admin connected from 127.0.0.1 using console on 2c21fe68dc17
admin@ncs-primary#
```

Step 28

Verify the status of the high availability setup with the **show high-availability** command.

Make sure that the primary node is connected to the secondary node. Secondary nodes are listed at the end of the output.

```
admin@ncs-primary# show high-availability status
high-availability status mode primary
high-availability status current-id nso-primary
high-availability status assigned-role primary
high-availability status read-only-mode false
ID             ADDRESS
---------------------------
nso-secondary  172.21.0.3

admin@ncs-primary#
```

Step 29

Enter the configuration mode and create a customer to test high availability replication. Then, exit the NSO CLI.

Use the following commands:

```
admin@ncs-primary# config
Entering configuration mode terminal
admin@ncs-primary(config)# customers customer ACME rank 1 status active
admin@ncs-primary(config-customer-ACME)# commit
Commit complete.
admin@ncs-primary(config-customer-ACME)# top
```

```
admin@ncs-primary(config)# exit
admin@ncs-primary# exit
student@nso-server:~/nso-ha$
```

Step 30

Enter the NSO CLI of the secondary node and display the customer information.

The customer that you created on the primary node has been replicated to the secondary node.

```
student@nso-server:~/nso-ha$ make testenv-cli NSO=secondary
docker exec -it testenv-nso-ha-6.1-student-nso-secondary bash -lc
'ncs_cli -Cu admin'

admin connected from 127.0.0.1 using console on f0315c8e9f0b
admin@ncs-secondary# show running-config customers
customers customer ACME
 rank   1
 status active
!
admin@ncs-secondary#
```

Step 31

Try entering the config mode.

An error will be displayed, telling you that this node is in read-only mode. While the high availability setup is active, configuration changes can only be performed on the primary node.

```
admin@ncs-secondary# config
Aborted: node is in read-only mode
admin@ncs-secondary#
```

Step 32

Exit the NSO CLI.

Use the **exit** command.

```
admin@ncs-secondary# exit
student@nso-server:~/nso-ha$
```

Activity Verification

You have completed this task when you attain these results:

- You have configured high availability on the nso-primary and nso-secondary nodes.
- Customer configuration has been successfully replicated between the nodes.

Task 2: Configure and Test a VIP Failover

Activity

Step 1

Enter the NSO CLI of the primary container and enter the config mode.

To enter the NSO CLI of the primary node, use the **make testenv-cli NSO=primary** command.

```
student@nso-server:~/nso-ha$ make testenv-cli NSO=primary
docker exec -it testenv-nso-ha-6.1-student-nso-primary bash -lc
'ncs_cli -Cu admin'

User admin last logged in 2024-02-08T20:51:30.842439+00:00, to
2c21fe68dc17, from 127.0.0.1 using cli-console
admin connected from 127.0.0.1 using console on 2c21fe68dc17
admin@ncs-primary# config
Entering configuration mode terminal
admin@ncs-primary(config)#
```

Step 2

Configure high availability VIP settings on the primary node and commit the changes.

Use **192.168.0.100** as the VIP address. Once committed, the configuration is automatically synced to the secondary node. Then, commit the changes and exit the CLI.

```
admin@ncs-primary(config)# hcc enabled vip-address 192.168.0.100
admin@ncs-primary(config)# commit
Commit complete.
admin@ncs-primary(config)# exit
admin@ncs-primary# exit
student@nso-server:~/nso-ha$
```

Step 3

Virtual IP **192.168.0.100** that you have configured should now be reachable and redirect to the primary.

Test the behavior from your lab machine over the RESTCONF interface with the **curl** tool. Call the **/restconf** URL on that particular IP and use the **admin** username and **admin** password credentials.

The response should contain an HTTP response code of 200.

```
student@nso-server:~/nso-ha$ curl -u admin:admin http://192.168.0.100/
restconf -v
*   Trying 192.168.0.100:80...
* Connected to 192.168.0.100 (192.168.0.100) port 80 (#0)
* Server auth using Basic with user 'admin'
> GET /restconf HTTP/1.1
> Host: 192.168.0.100
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 08 Feb 2024 20:58:42 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Length: 157
< Content-Type: application/yang-data+xml
< X-Cisco-NSO-Trace-ID: a61d4578-ea53-4d22-8f2f-8b46e67531bb
< Pragma: no-cache
< Content-Security-Policy: default-src 'self'; block-all-mixed-content;
base-uri 'self'; frame-ancestors 'none';
< Strict-Transport-Security: max-age=15552000; includeSubDomains
< X-Content-Type-Options: nosniff
< X-Frame-Options: DENY
< X-XSS-Protection: 1; mode=block
<
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <data/>
  <operations/>
  <yang-library-version>2019-01-04</yang-library-version>
</restconf>
* Connection #0 to host 192.168.0.100 left intact
student@nso-server:~/nso-ha$
```

Activity Verification

You have completed this task when you attain these results:

- You have configured a VIP 192.168.0.100, that redirects to the current primary node.
- An HTTP request towards RESTCONF API returns an HTTP status code 200.

Task 3: Test Failover Scenarios

Activity

Step 1

Simulate an outage of the primary host. Stop the NSO container that is hosting the primary node.

To simulate this action, display the list of currently running Docker containers and shut down the container that is hosting the primary NSO node, using the **docker stop** command.

```
student@nso-server:~/nso-ha$ docker ps -a
CONTAINER ID   IMAGE                                        COMMAND
CREATED              STATUS                         PORTS
NAMES
f0315c8e9f0b   nso303.gitlab.local/nso-ha/nso:6.1-student   "/run-
nso.sh"       About an hour ago   Up About an hour (healthy)   22/tcp,
80/tcp, 443/tcp, 830/tcp, 4334/tcp
testenv-nso-ha-6.1-student-nso-secondary
2c21fe68dc17   nso303.gitlab.local/nso-ha/nso:6.1-student   "/run-
nso.sh"       About an hour ago   Up About an hour (healthy)   22/tcp,
80/tcp, 443/tcp, 830/tcp, 4334/tcp
testenv-nso-ha-6.1-student-nso-primary
5b243c1b17f3   gitlab/gitlab-ee:13.4.0-ee                        "/assets/
wrapper"   3 months ago        Up 6 hours (healthy)         0.0.0.0:80-
>80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp,
0.0.0.0:8022->22/tcp, :::8022->22/tcp   gitlab_web_1
student@nso-server:~/nso-ha$ docker stop 2c21fe68dc17
2c21fe68dc17
student@nso-server:~/nso-ha$
```

Step 2

Enter the NSO CLI of the secondary node and display the high availability status on the secondary node.

The secondary node has now assumed the primary role in the high availability environment.

If the output on the secondary node does not match the output below, wait couple of seconds and retry again.

```
student@nso-server:~/lab/docker/nso-ha$ make testenv-cli NSO=secondary
docker exec -it testenv-nso-ha-6.1-student-nso-secondary bash -lc
'ncs_cli -Cu admin'

admin connected from 127.0.0.1 using console on f0315c8e9f0b
admin@ncs-secondary# show high-availability
high-availability enabled
high-availability status mode primary
high-availability status current-id nso-secondary
high-availability status assigned-role primary
high-availability status read-only-mode true
admin@ncs-secondary#
```

Step 3

Display the alarms with the **show alarms** command.

When a failover occurs, NSO raises an alarm to alert the operator about the new state of the system.

```
admin@ncs-secondary# show alarms
alarms summary indeterminates 0
alarms summary criticals 1
alarms summary majors 0
alarms summary minors 0
alarms summary warnings 0
alarms alarm-list number-of-alarms 1
alarms alarm-list last-changed 2024-02-08T21:09:02.690501+00:00
alarms alarm-list alarm ncs ha-primary-down /high-availability/ha-
```

```
node[id='nso-primary'] ""
 is-cleared              false
 last-status-change      2024-02-08T21:09:02.690501+00:00
 last-perceived-severity critical
 last-alarm-text         "Lost connection to primary due to: Primary
closed connection"
 status-change 2024-02-08T21:09:02.690501+00:00
  received-time      2024-02-08T21:09:02.690501+00:00
  perceived-severity critical
  alarm-text         "Lost connection to primary due to: Primary closed
connection"
admin@ncs-secondary#
```

Step 4

Exit the NSO CLI and try accessing the VIP RESTCONF URL again with the **curl** command.

You can see that the VIP address is still available, even though the original primary node is offline.

```
admin@ncs-secondary# exit
student@nso-server:~/nso-ha$ curl -u admin:admin http://192.168.0.100/
restconf
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <data/>
  <operations/>
  <yang-library-version>2019-01-04</yang-library-version>
</restconf>
student@nso-server:~/nso-ha$
```

Step 5

Turn the primary node container back on again and enter its NSO CLI after the container reaches healthy status.

To enter the NSO CLI of the primary node, use the **make testenv-cli NSO=primary** command.

```
student@nso-server:~/nso-ha$ docker ps -a
CONTAINER ID   IMAGE                                        COMMAND
CREATED              STATUS                         PORTS
NAMES
f0315c8e9f0b   nso303.gitlab.local/nso-ha/nso:6.1-student   "/run-
nso.sh"       About an hour ago   Up About an hour (healthy)   22/tcp,
80/tcp, 443/tcp, 830/tcp, 4334/tcp
testenv-nso-ha-6.1-student-nso-secondary
2c21fe68dc17   nso303.gitlab.local/nso-ha/nso:6.1-student   "/run-
nso.sh"       About an hour ago   Exited (143) 6 minutes ago
testenv-nso-ha-6.1-student-nso-primary
5b243c1b17f3   gitlab/gitlab-ee:13.4.0-ee                   "/assets/
wrapper"   3 months ago        Up 7 hours (healthy)         0.0.0.0:80-
>80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp,
0.0.0.0:8022->22/tcp, :::8022->22/tcp   gitlab_web_1
```

```
student@nso-server:~/nso-ha$ docker start 2c21fe68dc17
2c21fe68dc17
student@nso-server:~/nso-ha$ make testenv-cli NSO=primary
docker exec -it testenv-nso-ha-6.1-student-nso-primary bash -lc
'ncs_cli -Cu admin'

admin connected from 127.0.0.1 using console on 2c21fe68dc17
admin@ncs-primary#
```

Step 6

Check the high availability status.

Since the secondary node currently acts as a primary role, nso-primary serves as a secondary node.

```
admin@ncs-primary# show high-availability
high-availability enabled
high-availability status mode secondary
high-availability status current-id nso-primary
high-availability status assigned-role secondary
high-availability status be-secondary-result initialized
high-availability status primary-id nso-secondary
high-availability status read-only-mode false
```

Step 7

Switch the high availability roles back to normal on both nodes.

Use the **high-availability be-primary** command on the primary node, and **high-availability be-secondary-to node nso-primary** command on the secondary node. A node with a nominal role primary is also implicitly a failover-primary; it will act as failover-primary if its currently assigned role is a secondary.

```
admin@ncs-primary# high-availability be-primary
result ok
admin@ncs-primary# exit
student@nso-server:~/nso-ha$ make testenv-cli NSO=secondary
docker exec -it testenv-nso-ha-6.1-student-nso-secondary bash -lc
'ncs_cli -Cu admin'

admin connected from 127.0.0.1 using console on f0315c8e9f0b
admin@ncs-secondary# high-availability be-secondary-to node nso-primary
result Attempting to be secondary to node nso-primary
admin@ncs-secondary#
```

Step 8

Check the high availability status.

The high availability environment is now back to normal and the outage has been resolved.

```
admin@ncs-secondary# show high-availability status
high-availability status mode secondary
high-availability status current-id nso-secondary
high-availability status assigned-role secondary
high-availability status be-secondary-result initialized
high-availability status primary-id nso-primary
high-availability status read-only-mode false
admin@ncs-secondary#
```

Activity Verification

You have completed this task when you attain these results:

- You tested and remedied a failure scenario, the primary node failure.

When the high availability is enabled, where are connected secondary nodes are listed?

- ○ At the end of the **show high-availability status** command output on the primary node.
- ○ At the end of the **show high-availability status** command output on the secondary node.
- ○ At the end of the **show running-config high-availability** command output on the primary node.
- ○ At the end of the **show running-config high-availability** command output on the secondary node.