Managing TLS for Nexus Dashboard Data Broker as an App on Nexus Dashboard

This chapter contains the following details:

Generating TLS Self-Signed Certification Between NDB Server and NDB Switch for NXAPI in the App

This section describes how to generate TLS self-signed certification between NDB server and NDB Switch in the app deployment.You need to generate certificates and keys for each switch to enable TLS. TLS communication between NDBswitch and NDB server uses port 443 only.

Complete the following steps to generate TLS self-signed certification between NDB Server and NDB Switch for NXAPI:

Note You cannot configure a controller to communicate using port 80 after configuring TLS.

Generating Self-signed Certificate and Key

Use this procedure to generate self-signed certificate and key.

Before you begin

nsure that you have domain name configured on the switch using ip domain-name command for each NDB switch that acts as the Fully Qualified Domain Name (FQDN) for the switch. For example:

```
conf t
```

```
ip domain-name cisco.com hostname N9k-117
end
```

The FQDN for the switch is configured to N9K-117.cisco.com.

Step 1 Log in to one of the app containers as root user.

For logging in to an ND/ APIC container, see Logging in to a Container on Cisco Nexus Dashboard , on page 12 or Logging in to a Container on Cisco APIC , on page 12.

Step 2 Generate the private key and self-signed certificate using the **openssl req** command.

This command creates a certificate file (sw1-ca.pem) and a private key (sw1-ca.key).

```
docker@docker-virtual-machine:~/TLS$ openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out
sw1-ca.pem -outform PEM -keyout sw1-ca.key

Generating a 2048 bit RSA private key
...+++
...........................................+++
writing new private key to 'sw1-ca.key'

You are about to be asked to enter information that will be incorporated into your certificate
request.
What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few
fields but you can leave some blank
For some fields there will be a default value, If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA Locality Name (eg, city) []:SJ
Organization Name (eg, company) [Internet Widgits Pty Ltd]:cisco Organizational Unit Name (eg,
section) []:insbu
Common Name (e.g. server FQDN or YOUR name) []:N9K-117.cisco.com Email Address []:myname@cisco.com
```

Note If you have multiple switches, generate the certificate file and private key for each switch.

Step 3 Copy the certificate file, sw1-ca.pem, and keyfile, sw1-ca.key, to the switch using the **scp** command.

Example:

```
bash-4.2# scp sw1-ca.key admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key
100% 1704   992.7KB/s   00:00
4.6KB/s 00:00

bash-4.2# scp sw1-ca.pem admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key

100% 1704   992.7KB/s   00:00
4.6KB/s 00:00
```

Note If you have multiple switches, repeat this step for all the switches.

Step 4 Use the **cat** command to get the contents of sw1-ca.pem file; copy the contents of the same.

Create a file with the same name on all other containers and paste the copied contents into it using vi editor and save the changed file. Perform the same steps to copy the contents of `sw1-ca.key` file to all containers.

Step 5 Configure the certificate file, sw1-ca.pem, and keyfile, sw1-ca.key in the switch using the **nxapi** command.

Example:

```
N9K-117 (config)# nxapi certificate httpskey keyfile bootflash:sw1-ca.key
Upload done. Please enable. Note cert and key must match.
N9K-117 (config)# nxapi certificate httpscrt certfile bootflash:sw1-ca.pem
Upload done. Please enable. Note cert and key must match.
```

Note If you have multiple switches, configure the corresponding certificate and private key to each switch.

Step 6 Enable self-signed certificates on the switch using the **nxapi certificate** command.

Example:

```
N9K-117 (config)# nxapi certificate enable
```

Note Ensure that there is no error while enabling self-signed certificates on the switch.

Step 7 Log in to the containers of the app as root user.

Step 8 Copy and convert the sw1-ca.key and sw1-ca.pem files to .PEM format using the **copy** command.

Example:

```
cp sw1-ca.key sw1-ndb-privatekey.pem
cp sw1-ca.pem sw1-ndb-cert.pem
```

Step 9 Concatenate the private key and the certificate file using **cat**command.

Example:

```
docker@docker-virtual-machine:~/TLS$ cat sw1-ndb-privatekey.pem sw1-ndb-cert.pem > sw1-ndb.pem
```

Step 10 Convertthe .pem file to .p12 file format using the **openssl**command. Enter the export password when prompted to create a password protected .p12 certificate file.

Example:

```
docker@docker-virtual-machine:~/TLS$openssl pkcs12 -export -out sw1-ndb.p12 -in sw1-ndb.pem
Enter Export Password: cisco123
Verifying - Enter Export Password: cisco123
Enter a password at the prompt. Use the same password that you entered in the previous Step (cisco123
```

Step 11 Convert the sw1-ndb.p12 to a password protected Java KeyStore (tlsKeyStore) file using the **keytool** command. Use the `jre/bin` from the installed java directory.

Example:

```
docker@docker-virtual-machine:~/TLS$ ./(relativePath)/keytool -importkeystore -srckeystore sw1-ndb.p12
-srcstoretype pkcs12 -destkeystore tlsKeyStore -deststoretype jks
Enter Destination Keystore password:cisco123 Re-enter new password:cisco123
Enter source keystore password:cisco123 Entry for alias 1 successfully imported.
Import command completed: 1 enteries successfully imported, 0 enteries failed or cancelled.
```

Note By default an alias named "1" is stored in tlsKeyStore for the first switch. If the NDB controller is managing multiple switches, repeat this step for all the switches.When you add the second switch, the utility allows you to rename the first switch alias and also provides a provision to rename alias for the second switch. Refer examples as shown below.

```
keytool -importkeystore -srckeystore sw2-ndb.p12 -srcstoretype pkcs12 -destkeystore
tlsKeyStore -deststoretype jks
keytool -importkeystore -srckeystore sw3-ndb.p12 -srcstoretype pkcs12 -destkeystore
tlsKeyStore -deststoretype jks
```

Step 12 List and verify content in the java tlsKeyStore using the **keytool** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsKeyStore | more
```

Note Repeat steps 7 to 12 on all containers.

Creating the TLS TrustStore File

TrustStore is created from the self-signed certificates that are generated for one or more switches. It holds certificates for one or more switches in the controller. This section describes how to create a Truststore using the self-signed certificate created in Generating Self-Signed Certificate and Key section. If you have multiple switches in the controller, each switch will have separate certificate file (for example, sw1-ndb-cert.pem, sw2-ndb-cert.pem).

Use this procedure to create a TLS TrustStore file.

Note Perform this procedure on all app containers.

Step 1 Log in to the app container as root user.

For logging in to an ND/ APIC container, see Logging in to a Container on Cisco Nexus Dashboard , on page 12 or Logging in to a Container on Cisco APIC , on page 12.

Step 2 Convert the certificate file, such as, sw1-ndb-cert.pem to a Java TrustStore (tlsTrustStore) file using the keytool command. Enter a password when prompted to create a password protected Java TrustStore (tlsTrustStore) file. The password should be at least six characters. Use the jre/bin installed in the java directory.

Example:

```
docker@docker-virtual-machine:~/TLS$ ./(relativePath)/keytool -import -alias sw1 -file sw1-ndb-cert.pem
-keystore tlsTrustStore
Enter Export Password: cisco123
Verifying - Enter Export Password: cisco123
Enter a password at the prompt. Use the same password that you entered in the previous Step (cisco123)
```

If a NDB controller manages multiple switches, repeat this step for all the switches to add all switch keys into the same TrustStore. For example:

```
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw2 -file sw2-ndb-cert.pem
-keystore tlsTrustStore
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw3 -file sw3-ndb-cert.pem
-keystore tlsTrustStore
// Here sw2 and sw3 are alias for switch 2 and switch 3 for identification purpose.
```

Step 3 List and verify keys for multiple switches in the same tlsTrustStore using the keytool command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsTrustStore | more
```

Starting Nexus Dashboard Data Broker with TLS

Use this procedure for starting Nexus Dashboard Data Broker with TLS.

Step 1 Log in to the app container as root user.

For logging in to an ND/ APIC container, see Logging in to a Container on Cisco Nexus Dashboard , on page 12 or Logging in to a Container on Cisco APIC , on page 12.

Step 2 Copy the tlsKeystore and tlsTruststore files that you created to the configuration folder of the data broker. `(/home/app/ndb/configuration - for ND and /home/app/local-data/configuration - for APIC).`

Example:

```
for ND:
cp tlskeystore /home/app/ndb/configuration
cp tlsTrustStore /home/app/ndb/configuration

for APIC:
cp tlskeystore /home/app/local-data/configuration
cp tlsTrustStore /home/app/local-data/configuration
```

Note Perform steps 1 and 2 on all the app containers.

Step 3 Restart the app from the app tile on the host.

Configuring TLS KeyStore and TrustStore Passwords on Nexus Dashboard Data Broker

You need to configure TLS KeyStore and TrustStore passwords to enable Nexus Dashboard Data Broker to read password protectedTLS KeyStore and TrustStore files. To configure TLS KeyStore and TrustStore passwords on Nexus Dashboard Data Broker, complete these steps:

Note Perform this procedure on all the containers.

Step 1 Log in to the app container as root user.

For logging in to an ND/ APIC container, see Logging in to a Container on Cisco Nexus Dashboard , on page 12 or Logging in to a Container on Cisco APIC , on page 12.

Step 2 Navigate to `bin` directory.

Example: `cd /home/app/ndb/bin`

Step 3 Configure the TLS KeyStore and TrustStore passwords using the **ndb config-keystore-passwords** command.

Example:

```
./ndb config-keystore-passwords --user admin --password admin --url https://localhost:8443
--verbose --prompt --keystore-password cisco123 --truststore-password cisco123
```

When this command prompts for a password, enter *admin*.

After the TLS is enabled on NDB, all the connections between NDB server and NDB switch are established using port 443. Ensure that you change device connections in NDB to use port 443.

After successfully completing these steps, you can add nexus switch in the controller using port 443. Use FQDN of the switch to add the device to the Nexus Dashboard Data Broker controller. You can verify the Certificate information using the WebUI Sandbox of the switch.

Generating TLS 3rd Party Certification Between NDB Server and NDB Switch for NXAPI in the App

This section describes how to generate TLS 3rd party certification between NDB server and NDB Switch.You need to request for a separate certificate and key for each switch in you network. TLS communication between NDB switch and NDB server uses port 443 only.

Obtaining Certificates from a Certification Authority

You can obtain certificate from a Certification Authority (CA) in two ways. You can either directly approach a CA for both the private key and certificate. The CA will generate a private key on your behalf along with the certificate that contains the public key with issuing CA's signature.

In the other approach, you can generate a private key using tools such as, openssl and generate a Certificate Signing Request (CSR) to a certificate issuing authority. The CA generates the certificates with public key using the user identity information from CSR.

Before you begin

Ensure that you have domain name configured in the switch using ip domain-name command for each NDB switch that acts as the Fully Qualified Domain Name (FQDN) for the switch. For example:

```
conf t
```

```
ip domain-name cisco.com hostname N9k-117
end
```

The FQDN for the switch is configured to N9K-117.cisco.com.

Step 1 Log in to one of the app containers as root user.

For logging in to an ND/ APIC container, see Logging in to a Container on Cisco Nexus Dashboard , on page 12 or Logging in to a Container on Cisco APIC , on page 12.

Step 2 Generate the private key (cert.key) and certificate signing request (cert.req) using **openssl** command.

Example:

```
docker@docker-virtual-machine:~/Mallik/TLS_CA$ openssl req -newkey rsa:2048 -sha256 -keyout cert.key
-keyform PEM -out cert.req -outform PEM

Generating a 2048 bit RSA private key
...............+++
....................+++
writing new private key to 'cert.key'
Enter PEM pass phrase:  cisco123 Verifying - Enter PEM pass phrase: cisco123

You are about to be asked to enter information that will be incorporated into your certificate
request.
What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few
fields but you can leave some blank

For some fields there will be a default value, If you enter '.', the field will be left blank.

Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:CA Locality Name (eg, city) [Newbury]:SJ
Organization Name (eg, company) [My Company Ltd]:cisco Organizational Unit Name (eg, section)
[]:insbu
Common Name (eg, your name or your server's hostname) []:N9K-117.cisco.com Email Address
[]:myname@cisco.com

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:   cisco123 An optional company name []: cisco123


docker@docker-virtual-machine: # ls
cert.key cert.req
```

Step 3 Verify the CSR using the **openssl** command.

Example:

```
docker@docker-virtual-machine:~/Mallik/TLS_CA$ openssl req -noout -text -in cert.req
```

Step 4 The private key is generated with a security passphrase. You may need to unencrypt the private key. To remove the passphrase from the private key, use the **openssl** command.

Example:

```
docker@docker-virtual-machine:~/Mk/TLS_CA$ ls
cert.key cert.req
docker@docker-virtual-machine:~/Mk/TLS_CA$cp cert.key cert.keybkp
docker@docker-virtual-machine:~/Mk/TLS_CA$ rm cert.key
docker@docker-virtual-machine:~/Mk/TLS_CA$ openssl rsa -in cert.keybkp -out cert.key
```

```
Enter pass phrase for cert.keybkp: cisco123
```

Note Repeat this step to remove passphrase from private keys for all the switches.

Depending on the tier of the CA you choose, you can get up to three certificates (certificate chain) for each CSR. This means you get three certificates (root, intermediate and domain) from CA for each NDB switch. You need to check with CA to identify each type of certificate. Certificate naming convention might be different for different certifying authorities. For example: test-root-ca-2048.cer (root), test-ssl-ca.cer (intermediate), N9K-117.cisco.com.cer (domain).

Certificates are mostly shared in .PEM file format. The cert.req file data needs to be submitted to 3rd party certification authority. Follow the relevant procedures and get the three (certificate) files.

Step 5 Create a single certificate file from the three certificate files using the **cat** command. The concatenation should be done in the following order, domain certificate, root certificate, and intermediate certificate. Syntax for **cat** command: *cat domain certificateroot certificateintermediate certificate* > *server.cer*

Example:

```
$cat N9K-117.cisco.com.cer test-root-ca-2048.cer test-ssl-ca.cer > server.cer
```

Step 6 Edit the newly created server.cer file to separate the concatenated END and BEGIN lines. Do not delete anything in the file.

Example:

```
-----END CERTIFICATE----------BEGIN CERTIFICATE-----


////// Modify the above line like this by adding a line feed between the two.
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
```

Note Repeat this step all the switches.

Step 7 Copy the private key (cert.key) and the certificate from CA (server.cer) to the switch using the copy command.

Example:

```
bash-4.2# scp server.cer admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key
                                                              100% 1704   992.7KB/s   00:00
4.6KB/s 00:00

bash-4.2# scp cert.key admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key

100% 1704   992.7KB/s   00:00
4.6KB/s 00:00
```

Note Repeat this step all the switches.

Step 8 Use the cat command to get the contents of server.cer file; copy the contents of the same.

Create a file with same name on all other containers and paste the copied contents into it using vi editor and save the changed file.

Perform the same steps to copy the contents of cert.key file to all containers.

Step 9 Configure the certificate file, sw1-ca.pem, and keyfile, sw1-ca.key in the switch using the **nxapi** command.

Example:

```
N9K-117 (config)# nxapi certificate httpskey keyfile bootflash:cert.key Upload done. Please enable.
 Note cert and key must match. N9K-117 (config)#
N9K-117 (config)# nxapi certificate httpscrt certfile bootflash:server.cer
Upload done. Please enable. Note cert and key must match.
```

Note If you have multiple switches, configure the corresponding certificate and private key to each switch.

Step 10 Enable self-signed certificates on the switch using the nxapi certificate command.

Example:

```
N9K-117 (config)# nxapi certificate enable N9K-117
```

Note Ensure that there is no error while enabling self-signed certificates on the switch.

Creating TLS KeyStore and TrustStore Files for Nexus Dashboard Data Broker Controller

Nexus Dashboard Data Broker uses certificates and keys to secure communication between switches. It stores the keys and certificates in keystores. These files are stored as tlsTruststore and tlsKeystore files in Nexus Dashboard Data Broker.

Use this procedure to generate the Java tlsKeyStore and tlsTrustStore files.

Step 1 Create a TLS directory and navigate to it.

Example:

```
mkdir -p TLS cd TLS
```

Step 2 Create three directories under mypersonalca and two prerequisite files.

Example:

```
mkdir -p mypersonalca/certs
mkdir -p mypersonalca/private
mkdir -p mypersonalca/crl
echo "01" > mypersonalca/serial
touch mypersonalca/index.txt
```

Generate the TLS private key and Certification Authority (CA) files for each switch connected to Nexus Dashboard Data Broker using the command.

Step 3 Copy the cert.key and server.cer created in the Obtaining Certificates from a Certification Authority section into the current directory (TLS). Select the certificate and key files for a single switch. These files were earlier generated for all the switches connecting to the controller. Using the server.cer and cert.key for the current switch, create the TLS KeyStore File.

If multiple switches are connected, repeat this step for each switch separately.

Step 4 Copy and convert the server.cer and cert.key files to .PEM format using the **copy** command.

Example:

```
cp cert.key sw1-ndb-privatekey.pem
cp server.cer sw1-ndb-cert.pem
```

Step 5 Copy the .pem files generated in the above step into all the containers.

Step 6 Concatenate the private key (sw1-ndb-privatekey.pem) and certificate file (sw1-ndb-cert.pem) into a single .PEM file using the **cat** command.

Example:

```
cat sw1-ndb-privatekey.pem sw1-ndb-cert.pem > sw1-ndb.pem
```

Step 7 Convertthe .PEM file to .P12 format using the **openssl** command. Enter the export password when prompted. The password must contain at least 6 characters, for example, cisco123. The sw1-ndb.pem file is converted to a password-protected sw1-ndb.p12 file.

Example:

```
docker@docker-virtual-machine:~/TLS$openssl pkcs12 -export -out sw1-ndb.p12 -in sw1-ndb.pem
Enter Export Password: cisco123
Verifying - Enter Export Password: cisco123
Enter a password at the prompt. Use the same password that you entered in the previous Step (cisco123)
```

Step 8 Convertthe sw1-ndb.p12 to a password protected Java KeyStore (tlsKeyStore) file using the **keytool** command. This command converts the sw1-ndb.p12 file to a password-protected tlsKeyStore file.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -importkeystore -srckeystore sw1-ndb.p12 -srcstoretype
 pkcs12 -destkeystore tlsKeyStore -deststoretype jks
Enter Destination Keystore password:cisco123
```

Note Bydefault an alias named “1” is stored in tlsKeyStore for the first switch. If the Nexus Dashboard Data Broker controller is managing multiple switches, repeat this step for all the switches. When you add the second switch, the utility allows you to rename the first switch alias and also provides a provision to rename alias for the new switch. For example, see below.

```
keytool -importkeystore -srckeystore sw2-ndb.p12 -srcstoretype pkcs12 -destkeystore
tlsKeyStore -deststoretype jks
keytool -importkeystore -srckeystore sw3-ndb.p12 -srcstoretype pkcs12 -destkeystore
tlsKeyStore -deststoretype jks
```

Step 9 List and verify content in the java tlsKeyStore using the **keytool** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsKeyStore | more
```

Step 10 Convertthe certificate file (sw1-ndb-cert.pem) to a Java TrustStore (tlsTrustStore) file using the **keytool** command. Enter a password when prompted to create a password protected Java TrustStore (tlsTrustStore) file. The password should be at least six characters.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw1 -file sw1-ndb-cert.pem -keystore
tlsTrustStore
Enter keystore password: cisco123 Re-enter new password: cisco123
Owner: EMAILADDRESS=myname@cisco.com, CN=localhost, OU=insbu, O=cisco, L=SJ, ST=CA, C=US Issuer:
EMAILADDRESS=myname@cisco.com, CN=localhost, OU=insbu, O=cisco, L=SJ, ST=CA, C=US Serial number:
c557f668a0dd2ca5
Valid from: Thu Jun 15 05:43:48 IST 2017 until: Sun Jun 13 05:43:48 IST 2027 Certificate fingerprints:
MD5: C2:7B:9E:26:31:7A:74:25:55:DF:A7:91:C9:5D:20:A3
SHA1: 3C:DF:66:96:72:12:CE:81:DB:AB:58:30:60:E7:CC:04:4D:DF:6D:B2 SHA256:
DD:FB:3D:71:B4:B8:9E:CE:97:A3:E4:2D:D3:B6:90:CD:76:A8:5F:84:77:78:BE:49:6C:04:01:84:62:2C:2F:EB
Signature algorithm name: SHA256withRSA Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false AuthorityKeyIdentifier [
KeyIdentifier [
0000: 0D B3 CF 81 66 4A 33 4E EF 86 7E 26 C3 50 9B 73 ....fJ3N...&.P.s
0010: 38 EF DF 40 8..@
]
]

#2: ObjectId: 2.5.29.19 Criticality=false BasicConstraints:[
CA:true PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false SubjectKeyIdentifier [
KeyIdentifier [
0000: 0D B3 CF 81 66 4A 33 4E EF 86 7E 26 C3 50 9B 73 ....fJ3N...&.P.s
0010: 38 EF DF 40 8..@
]
]

Trust this certificate? [no]: yes Certificate was added to keystore
```

If a Nexus Dashboard Data Broker controller manages multiple switches, repeat this step for all the switches to add all switch keys into the same TrustStore. Example is as shown below:

```
keytool -import -alias sw2 -file sw2-ndb-cert.pem -keystore tlsTrustStore
keytool -import -alias sw3 -file sw3-ndb-cert.pem -keystore tlsTrustStore
```

Step 11 List and verify keys for multiple switches in the same tlsTrustStore using the **keytool** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsTrustStore | more
```

Note Perform steps 6 to 11 on all the containers.

What to do next

Start Nexus Dashboard Data Broker using TLS. See Starting Nexus Dashboard Data Broker with TLS , on page 5 for the detailed procedure.

After that, configure TLS KeyStore and TrustStore passwords. See Configuring TLS KeyStore and TrustStore Passwords on Nexus Dashboard Data Broker , on page 5 for the detailed procedure.

Logging in to a Container on Cisco Nexus Dashboard

Use this procedure to log in to a container on Cisco Nexus Dashboard (ND).

Step 1 Login to any of the ND node as root user, using ssh and the generated temporary root password.

```
ssh root@10.16.206.50
root@10.16.206.50's password: <enter temporary root password here>
[root@ND-1 ~]#
```

Step 2 The available pods related to NDDB are listed.

Example:

```
[root@ND-1 ~]# kubectl -n cisco-ndb get pods
NAME           READY   STATUS    RESTARTS   AGE
ndbserver-0    1/1     Running   0          12d
ndbserver-1    1/1     Running   0          12d
ndbserver-2    1/1     Running   0          12d
```

Step 3 Get a bash shell to the container running in any of the pods using the **kubectl -n cisco-ndb exec -it <pod-name> -- bash** command.

Example:

```
[root@ND-1 ~]# kubectl -n cisco-ndb exec -it ndbserver-0 -- bash
```

Logging in to a Container on Cisco APIC

Use this procedure to log in to a container on Cisco APIC.

Step 1 Login to any of the APIC nodes as root user, using ssh and the generated temporary root password.

```
ssh root@10.16.206.247
Application Policy Infrastructure Controller
root@10.16.206.247's password: <enter temporary root password here>
```

Step 2 The NDDB container running on that node is listed.

Example:

```
[root@apic1 ~]# docker ps | grep ndb
2bb5309cbbae  31d3a284752b  "/opt/bin/conit.bi..."  5 days ago  Up 5 days
ndbserver-c1cf9a4d-ab31-06d3-b8c0-b01840fb6cc9
```

Step 3 Get a bash shell to that container using the **docker exec -it <container-id> /bin/bash** command.

Example:

```
[root@apic1 ~]# docker exec -it ndbserver-c1cf9a4d-ab31-06d3-b8c0-b01840fb6cc9 /bin/bash
```