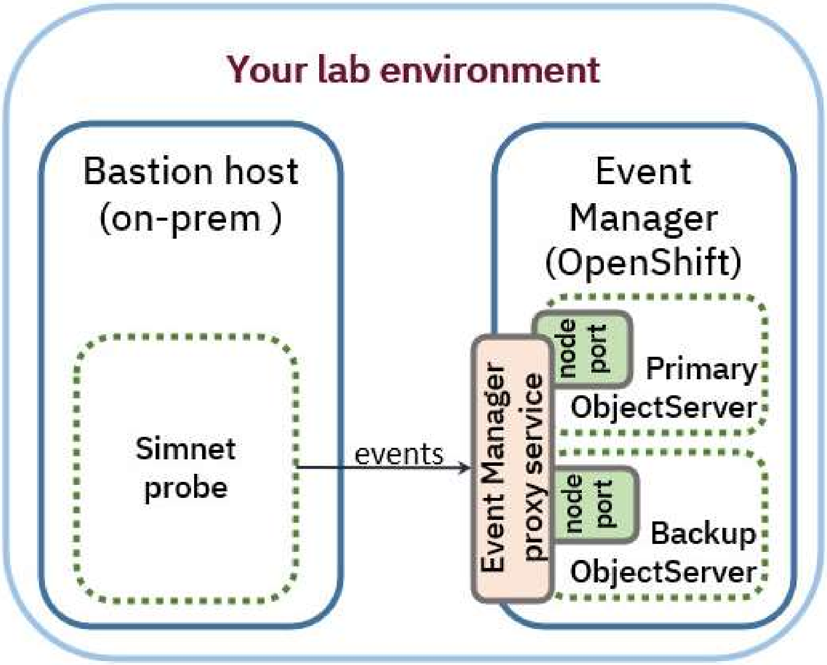
Section 1. Connecting an on-premises probe to Event Manager running in Red Hat OpenShift

You begin this section by exposing the Event Manager ObjectServer databases running in your OpenShift cluster to external traffic. You then configure the Simnet probe, running “on-prem” on your bastion host to connect to Event Manager. The Simnet probe is a special test probe that is used to generate simulated events. At the end of the section, you start the probe and confirm data is flowing from your on-prem probe to Event Manager.

Configuring Event Manager to use node port services

One way to expose a service running within a Red Hat OpenShift cluster to external network traffic is to use node ports. Node ports expose a static port on each node. The port number assigned on each node is randomly selected from the range 30000-32767 when the service is created.



Before an on-prem probe can send events to Event Manager running in Red Hat OpenShift, you must expose the primary and backup ObjectServer services. These two ObjectServers run the central data stores within Event Manager, including the database where all events are stored. On-prem probes must be able to connect to these ObjectServers to send events.

Internally, access to the ObjectServers is through a proxy service. In this section you configure the Event Manager proxy service to use node port as the service type.

Important

Many of the steps in this exercise direct you to edit Red Hat OpenShift resources directly, using the `oc edit <target>` command. When you run the `oc edit <target>` command, the target resource opens in your default text editor, which is gedit.

It is important to note that when you edit the YAML configuration of a Red Hat OpenShift resource directly with the `oc edit` command, you are editing a live document. Follow these guidelines when using gedit to edit a live YAML configuration file:

- Close the gedit text editor before you run the `oc edit <target>` command. The text editor will open automatically when you run the command.
- Save your changes to the YAML document only once, after you have finished all of your edits. Each time you save the YAML document, OpenShift adjusts settings within one or more live resources. To avoid unexpected results, save the file only one time.
- After you have saved your changes, close the document and close the text editor.
- If you save the document with mistakes in it, such as typos or improper syntax, all of your changes are discarded by OpenShift. In this case, you must open the document again and reenter your changes correctly.

__ 1. Open a terminal window if you do not already have one open.

__ 2. Use the following command to log in to your OpenShift cluster if you are not already logged in.

```
oc login -u ocadmin -p ibmocp48
```

Example output:

```
Login successful.

You have access to 65 projects, the list has been suppressed. You can list all
projects with 'oc projects'

Using project "noi".
```

__ 3. Change to the noi namespace, if you are not already working with the noi namespace.

```
oc project noi
```

__ 4. If you have a text editor open, close it.

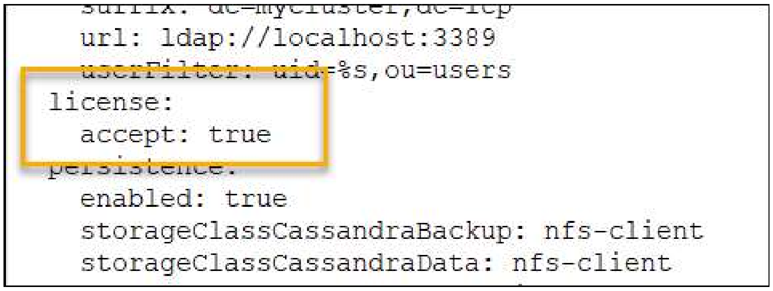
__ 5. Edit your Event Manager instance to enable node port access.

__ a. Run the following command to edit your Event Manager instance. This action opens a YAML configuration in a text editor.

```
oc edit noi evtmanager
```

__ b. Find the following lines in the file. You can safely ignore the color of the text in the document. The text in your lab environment might be colored by the text editor, or it might be black and white, like in this example.

```
license:
    accept: true
```



```
    suffix: dc=mycluster,dc=icp
    url: ldap://localhost:3389
    userFilter: uid=%s,ou=users
  license:
    accept: true
  persistence:
    enabled: true
    storageClassCassandraBackup: nfs-client
    storageClassCassandraData: nfs-client
```

__ c. Add the following two lines under the `accept: true` line. Make sure that the two lines you add are indented to the exact level as the two lines above them. This file is indented with spaces, not tabs.

Indent the line `helmValuesNOI:` two spaces from the left margin of the file.

Indent the line `global.service.nodePort.enable: true` four spaces from the left margin of the file.

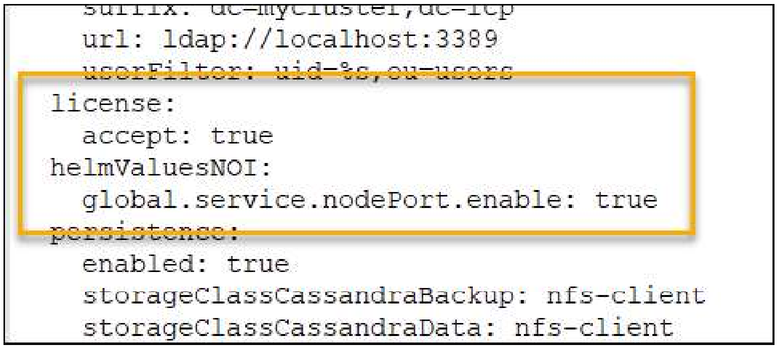
```
helmValuesNOI:
  global.service.nodePort.enable: true
```

Hint

You can find a plain-text version of this example, as well as other examples used during your labs, in the file: `/home/netcool/ClassFiles/longCodeExamples.txt`.

__ d. Confirm that your file looks like the following example. You could have added the two new lines anywhere in the `spec:` section of the file, but the license acceptance lines are a convenient point of reference.

```
  license:
    accept: true
  helmValuesNOI:
    global.service.nodePort.enable: true
```



```
    suffix: dc=mycluster,dc=icp
    url: ldap://localhost:3389
    userFilter: uid=%s,ou=users
  license:
    accept: true
  helmValuesNOI:
    global.service.nodePort.enable: true
  persistence:
    enabled: true
    storageClassCassandraBackup: nfs-client
    storageClassCassandraData: nfs-client
```

__ e. Save the file.

__ f. Close the text file and the text editor.

Note

After you save and close the file, the two ObjectServer pods, named `evtmanager-ncobackup-0` and `evtmanager-ncoprimary-0` will become unavailable. In the next steps, you restart these two pods and they will operate normally again.

__ 6. Run the following command to verify that your changes were saved. Be sure that the output of the command matches the following example before you go to the next step.

```
oc describe noi evtmanager | grep "Helm Values NOI" -A1
```

Example output:

```
  Helm Values NOI:
    global.service.nodePort.enable:  true
```

Troubleshooting

If you do not see the expected output in the preceding command, go back to step 5, where you edit your Event Manager instance configuration. Make sure you added the two new lines of configuration correctly.

__ 7. Edit the primary ObjectServer stateful set to correct the host name that the ObjectServer container is listening on.

__ a. Find the name of the primary ObjectServer stateful set.

```
oc get statefulset | grep primary
```

Example output:

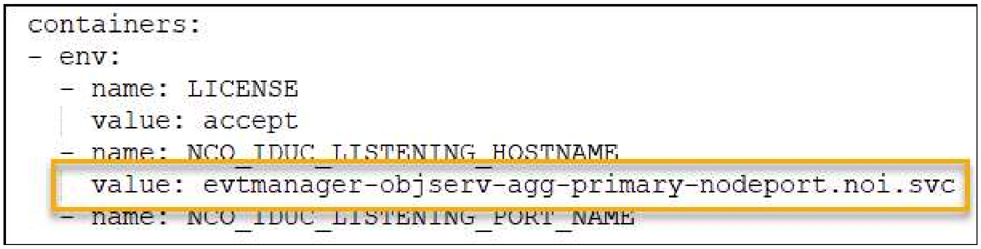
```
evtmanager-ncoprimary               0/1     3d22h
```

__ b. Edit the primary ObjectServer stateful set. This action opens a YAML configuration in a text editor.

```
oc edit statefulset evtmanager-ncoprimary
```

__ c. Find the following line in the file:

```
value: evtmanager-objserv-agg-primary-nodeport.noi.svc
```

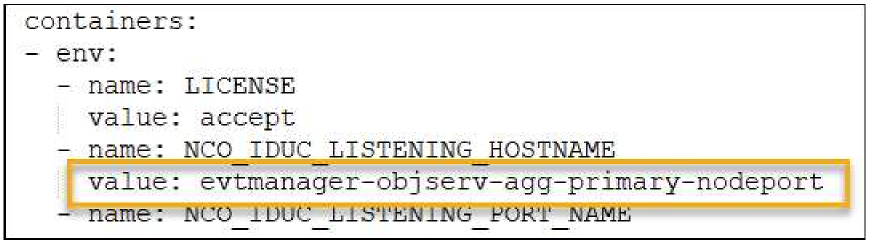


```
containers:
- env:
  - name: LICENSE
    value: accept
  - name: NCO_IDUC_LISTENING_HOSTNAME
    value: evtmanager-objserv-agg-primary-nodeport.noi.svc
  - name: NCO_IDUC_LISTENING_PORT_NAME
```

__ d. Remove the `.noi.svc` suffix at the end of the line.

__ e. Confirm that your line looks like the following example.

```
value: evtmanager-objserv-agg-primary-nodeport
```



```
containers:
- env:
  - name: LICENSE
    value: accept
  - name: NCO_IDUC_LISTENING_HOSTNAME
    value: evtmanager-objserv-agg-primary-nodeport
  - name: NCO_IDUC_LISTENING_PORT_NAME
```

__ f. Save the file.

__ g. Close the text file and the text editor.

__ 8. Edit the backup ObjectServer stateful set to correct the host name that the backup ObjectServer container is listening on.

__ a. Find the name of the backup ObjectServer stateful set.

```
oc get statefulset | grep backup
```

Example output:

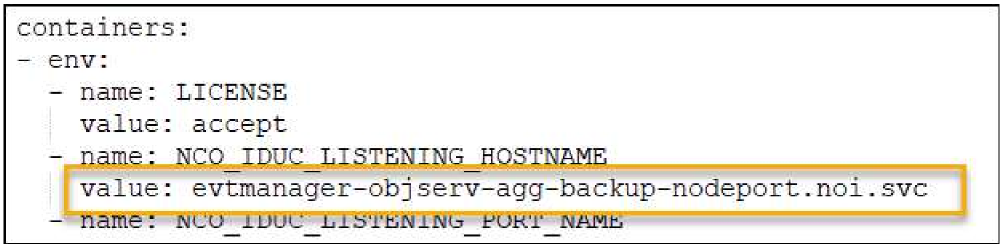
```
evtmanager-ncobackup                0/1     3d22h
```

__ b. Edit the backup ObjectServer stateful set. This action opens a YAML configuration in a text editor.

```
oc edit statefulset evtmanager-ncobackup
```

__ c. Find the following line in the file:

```
value: evtmanager-objserv-agg-backup-nodeport.noi.svc
```

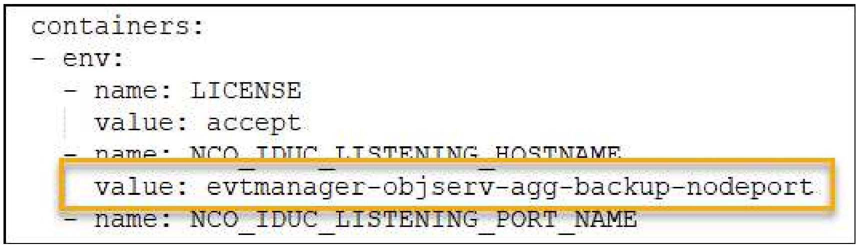


```
containers:
- env:
  - name: LICENSE
    value: accept
  - name: NCO_IDUC_LISTENING_HOSTNAME
    value: evtmanager-objserv-agg-backup-nodeport.noi.svc
  - name: NCO_IDUC_LISTENING_PORT_NAME
```

__ d. Remove the `.noi.svc` suffix at the end of the line.

__ e. Confirm that your line looks like the following example.

```
value: evtmanager-objserv-agg-backup-nodeport
```



```
containers:
- env:
  - name: LICENSE
    value: accept
  - name: NCO_IDUC_LISTENING_HOSTNAME
    value: evtmanager-objserv-agg-backup-nodeport
  - name: NCO_IDUC_LISTENING_PORT_NAME
```

__ f. Save the file.

__ g. Close the text file and the text editor.

__ 9. Restart the primary and backup ObjectServer pods

__ a. Run the following command to restart the primary and backup ObjectServer pods.

```
oc get pods | egrep "nco.*-0" | awk '{ print $1}' | xargs oc delete pod
```

Example output:

```
pod "evtmanager-ncobackup-0" deleted
pod "evtmanager-ncoprimary-0" deleted
```

__ b. After a few moments, run the following command to check if the pods restarted and are running correctly. If either pod is not running, wait a few moments and try the command again.

```
oc get pods | egrep "nco.*-0"
```

Example output:

```
evtmanager-ncobackup-0          2/2     Running     0          78s
evtmanager-ncoprimary-0         1/1     Running     0          72s
```

__ 10. Verify your changes to the Event Manager proxy service and find your ObjectServer port numbers.

__ a. Run the following command to show the configuration of the Event Manager proxy service, which provides access to the ObjectServer pods. Notice the following details about the Event Manager proxy configuration:

- The primary ObjectServer uses the node port named `aggp-proxy-port`, which uses port `32391`.
- The backup ObjectServer uses the node port named `aggb-proxy-port`, which uses port `30851`.
- The Event Manager proxy service `type` is `NodePort`.

```
oc get service -o yaml evtmanager-proxy -n noi
```

Example output:

```
...output omitted...
ports:
  - name: aggp-proxy-port
    nodePort: 32391
    port: 6001
    protocol: TCP
    targetPort: 6001
  - name: aggb-proxy-port
    nodePort: 30851
    port: 6002
    protocol: TCP
    targetPort: 6002
  selector:
    app.kubernetes.io/instance: evtmanager
    app.kubernetes.io/name: proxy
  sessionAffinity: None
  type: NodePort
...output omitted...
```

__ b. Write the two `nodePort` values down. The port values you need to write down are those in the range: 30000-32767. Make a note of which port goes to the primary ObjectServer and which port goes to the backup ObjectServer. Use the port numbers in your environment, not the port numbers in the example.

Important

Write the two `nodePort` values down. You need to know these two port numbers later in this exercise. Write down the port numbers in your environment, not the port numbers in the example.

__ 11. Test external access to the Event Manager proxy service.

To verify connectivity to the node ports from outside your cluster, you can run a test from the bastion host to the Event Manager proxy service. The bastion host is outside of the cluster and the Event Manager proxy is running inside of the cluster, so a connectivity test between them would verify external access. You use openssl to run this test in the next steps.

In your lab environment, all outside traffic coming into your OpenShift cluster is directed through an external load balancer. This load balancer is running on a host in your lab named `infra.labs.ihost.com`. In your test, you will send an openssl client request from the bastion host to the load balancer. The load balancer will direct the request to the compute nodes where the Event Manager proxy service is running. The Event Manager proxy service should respond with details about its TLS certificate. If the request is successful, that confirms that node port access is successfully configured.

__ a. Run the following command to test external access to the primary ObjectServer node port. If the output of your command is similar to the following example, the primary ObjectServer node port is configured correctly. You can ignore any error messages about self-signed certificates.

```
openssl s_client -showcerts -connect infra.labs.ihost.com:<YOUR_PRIMARY_PORT>
```

For example:

```
openssl s_client -showcerts -connect infra.labs.ihost.com:32391
```

Example output:

```
...output omitted...

    0050 - 40 a7 35 08 f5 73 c0 c1-06 13 10 44 ca b7 d1 c9   @.5..s.....D....
    0060 - fc 11 21 26 ac be a4 89-16 a5 2b 64 8b 6e ee dc   ..!&......+d.n..
    0070 - c1 40 c0 bc c4 cb cf c0-                          .@......

    Start Time: 1655392439
    Timeout   : 7200 (sec)
    Verify return code: 19 (self signed certificate in certificate chain)
    Extended master secret: no
```

__ b. Scroll up through the output of the command and find the subject of the certificate, which looks like the following example. This confirms that the response to your openssl request came from the Event Manager proxy service. Write down the `subject` value, which in this example is `evtmanager-proxy.noi.svc`. You need this value later in these exercises.

```
-----END CERTIFICATE-----
---
Server certificate
subject=CN = evtmanager-proxy.noi.svc

issuer=CN = openshift-service-serving-signer@1641526827
```

__ c. Press **Ctrl+c** to return to your command prompt.

__ d. Run the openssl test again, this time using the backup ObjectServer port.

```
openssl s_client -showcerts -connect infra.labs.ihost.com:<YOUR_BACKUP_PORT>
```

__ e. Confirm that the output of the command is similar to your first test. Press **Ctrl+c** to return to your command prompt.

Connecting an on-premises probe to Event Manager

Use these steps to configure and start the probe, then verify that data is flowing from the probe to Event Manager.

In your lab environment, all data going into your OpenShift cluster passes through an external load balancer first. The load balancer is running on the host named `infra.labs.ihost.com`. The IP address of this host is your cluster IP, because network traffic sent to that IP address is forwarded to the nodes in your cluster.

__ 1. Run the following command to find your cluster IP address. Write down the cluster IP address, you use it in the next step.

```
host infra.labs.ihost.com
```

Example output:

```
infra.labs.ihost.com has address 10.100.1.2
```

__ 2. Add an entry in the `/etc/hosts` file of the server where the probe runs. In your environment, the probe runs on the bastion host.

__ a. Run the following command to open the `/etc/hosts` file in a text editor.

```
sudo gedit /etc/hosts &
```

__ b. Add a line similar to the following example to the bottom of the file. Use the cluster IP address you found in the preceding step as the first part of the line.

Use the internal DNS name of the Event Manager proxy service as the second part of the line, for example: `evtmanager-proxy.noi.svc`.

```
10.100.1.2 evtmanager-proxy.noi.svc
```

__ c. Save the `/etc/hosts` file.

__ d. Close the text file and the text editor.

Hint

If you need to find the internal DNS name of your Event Manager proxy service again, use the following command. You can find the DNS name in the subject line of the certificate. In this example, the DNS name is `evtmanager-proxy.noi.svc`.

```
openssl s_client -showcerts -connect infra.labs.ihost.com:<YOUR_PRIMARY_PORT>
```

For example:

```
openssl s_client -showcerts -connect infra.labs.ihost.com:32391
```

Example output:

```
-----END CERTIFICATE-----
Server certificate
subject=CN = evtmanager-proxy.noi.svc

issuer=CN = openshift-service-serving-signer@1641526827
```

Note

You can ignore any `Set document metadata failed` messages in the terminal. These messages appear because you opened the text editor with sudo access.

__ 3. Connections from probes to ObjectServers running in OpenShift should always run in secure mode. Configure your probe to connect securely using the security certificate from the proxy service.

__ a. Change to the `/home/netcool/ClassFiles` directory.

```
cd /home/netcool/ClassFiles
```

__ b. Run the following command to download the TLS security certificate from the Event Manager proxy service to your bastion host, where the probe runs.

```
oc get secrets/signing-key -n openshift-service-ca -o \
template='{{index .data "tls.crt"}}' | base64 --decode > cluster-ca-cert.pem
```

Hint

You can find a plain-text version of this example, as well as other examples used during your labs, in the file: `/home/netcool/ClassFiles/longCodeExamples.txt`.

__ c. Run the following command to create a secure keystore for your probe.

```
$NCHOME/bin/nc_gskcmd -keydb -create -db  \
"$NCHOME/etc/security/keys/omni.kdb" -pw password -stash -expire 1000
```

__ d. Copy the TLS certificate you downloaded earlier to your new keystore.

```
$NCHOME/bin/nc_gskcmd -cert -add -file cluster-ca-cert.pem -db  \
$NCHOME/etc/security/keys/omni.kdb -stashed
```

__ e. Run the following command to verify that the TLS certificate is now in your keystore.

```
$NCHOME/bin/nc_gskcmd -cert -list -db \
$NCHOME/etc/security/keys/omni.kdb -pw password
```

Example output:

```
Certificates found
* default, - personal, ! trusted, # secret key
!   CN=openshift-service-serving-signer@1641526827
```

Note

The probe uses an interfaces file to map ObjectServer names to host names, however, you should not edit the interfaces file directly. A best practice is to edit the `omni.dat` file, then run the `nco_igen` tool to update your interfaces file based on the contents of `omni.dat`. The next steps guide you through this process.

__ 4. Update your `omni.dat` file to include the primary and backup ObjectServers running in your OpenShift cluster.

__ a. Open the `omni.dat` file in a text editor.

```
gedit $NCHOME/etc/omni.dat &
```

__ b. Add the following lines to the bottom of the file. Replace the values in angle brackets (<>) with the node port numbers you found in a previous step.

```
[OCP_AGG_P]
{
 Primary: evtmanager-proxy.noi.svc ssl <YOUR_PRIMARY_NODEPORT>
}
[OCP_AGG_B]
{
 Primary: evtmanager-proxy.noi.svc ssl <YOUR_BACKUP_NODEPORT>
}
[OCP_AGG_V]
{
 Primary: evtmanager-proxy.noi.svc ssl <YOUR_PRIMARY_NODEPORT>
 Backup: evtmanager-proxy.noi.svc ssl <YOUR_BACKUP_NODEPORT>
}
```

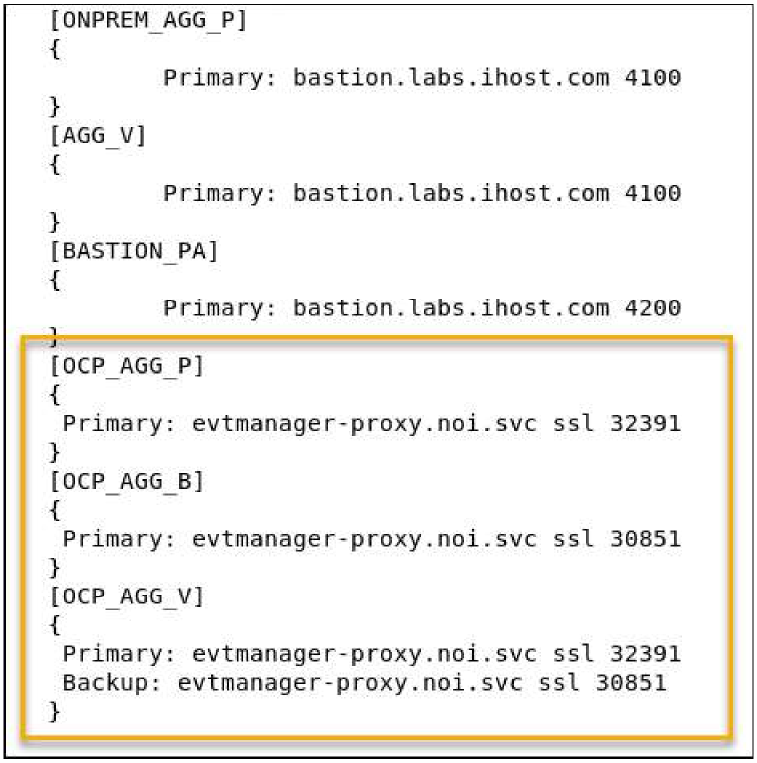
Hint

Use the following command to find the node port numbers for the primary and backup ObjectServers.

```
oc get service -o yaml evtmanager-proxy -n noi | grep -iB2 nodeport:
```

__ c. Confirm that your `omni.dat` file looks like the following example. Of course, use the actual node port numbers from your own environment, not from the example.

```
[ONPREM_AGG_P]
{
   Primary: bastion.labs.ihost.com 4100
}
[AGG_V]
{
   Primary: bastion.labs.ihost.com 4100
}
[BASTION_PA]
{
   Primary: bastion.labs.ihost.com 4200
}
[OCP_AGG_P]
{
 Primary: evtmanager-proxy.noi.svc ssl 32391
}
[OCP_AGG_B]
{
 Primary: evtmanager-proxy.noi.svc ssl 30851
}
[OCP_AGG_V]
{
 Primary: evtmanager-proxy.noi.svc ssl 32391
 Backup: evtmanager-proxy.noi.svc ssl 30851
}
```



```
[ONPREM_AGG_P]
{
        Primary: bastion.labs.ihost.com 4100
}
[AGG_V]
{
        Primary: bastion.labs.ihost.com 4100
}
[BASTION_PA]
{
        Primary: bastion.labs.ihost.com 4200
}
[OCP_AGG_P]
{
 Primary: evtmanager-proxy.noi.svc ssl 32391
}
[OCP_AGG_B]
{
 Primary: evtmanager-proxy.noi.svc ssl 30851
}
[OCP_AGG_V]
{
 Primary: evtmanager-proxy.noi.svc ssl 32391
 Backup: evtmanager-proxy.noi.svc ssl 30851
}
```

__ d. Save and close the omni.dat file.

__ e. Close the text editor.

__ 5. Run the nco_igen tool to update your interfaces file based on the contents of omni.dat.

```
$NCHOME/bin/nco_igen
```

__ 6. Test the connection from your bastion host to the ObjectServers running in OpenShift. Probes include a tool called nco_ping that tests connectivity using an SQL Insert, Delete, Update, and Command (IDUC) communication protocol running over TCP/IP.

__ a. Use the nco_ping tool to test connectivity to your primary ObjectServer.

```
nco_ping OCP_AGG_P
```

Example output:

```
NCO_PING: Server available.
```

__ b. Test connectivity to your backup ObjectServer.

```
nco_ping OCP_AGG_B
```

Example output:

```
NCO_PING: Server available.
```

__ c. Test connectivity to your virtual failover pair ObjectServer.

```
nco_ping OCP_AGG_V
```

Example output:

```
NCO_PING: Server available.
```

Important

Confirm that the output of each nco_ping test is: Server available. If any of the ObjectServers are not available, go back and confirm that your omni.dat file is correct, then run the nco_igen tool again to update the interfaces file.

__ 7. Configure your probe to connect to the virtual ObjectServer failover pair.

__ a. Probes are configured with a properties file. Make a copy of the probe's properties file before you edit it.

```
cp $OMNIHOME/probes/linux2x86/simnet.props $OMNIHOME/probes/linux2x86/simnet.props.ORIG
```

__ b. Open the properties file in a text editor.

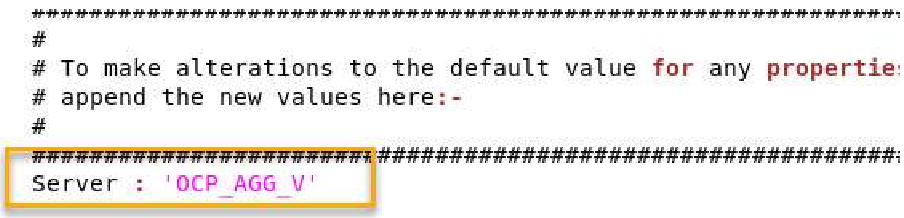
```
gedit $OMNIHOME/probes/linux2x86/simnet.props &
```

__ c. Add the following line to the bottom of the file. This line configures your probe to send events to the virtual ObjectServer failover pair.

```
Server : 'OCP_AGG_V'
```

__ d. Confirm that your `simnet.props` file looks like the following example.

```
...output ommited...
#######################################################################
#
# To make alterations to the default value for any properties
# append the new values here:-
#
#######################################################################
Server : 'OCP_AGG_V'
```



```
#################################################################:
#
# To make alterations to the default value for any propertie:
# append the new values here:-
#
#################################################################:
Server : 'OCP_AGG_V'
```

__ e. Save and close the `simnet.props` file.

__ f. Close the text editor.

__ 8. Start the probe.

__ a. Run the following command to start the probe and run it in the background.

```
$OMNIHOME/probes/nco_p_simnet &
```

Example output:

```
[1] 489307
...output omitted...
```

__ b. Use the following command to confirm that the probe is running. Look for the `nco_p_simnet` process.

```
ps -ef | grep nco
```

Example output:

```
netcool   488884    5352  1 09:03 pts/0    00:00:00
/opt/IBM/tivoli/netcool/omnibus/platform/linux2x86/probes64/nco_p_simnet
netcool   483924    5352  1 09:03 pts/0    00:00:00 netcool   491641    5352  0
10:29 pts/0    00:00:00 grep --color=auto nco
```

__ 9. Use the Event Manager WebGUI user interface to confirm that Event Manager is receiving events from the probe.

__ a. Log in to the Event Manager WebGUI user interface with the user name `icpadmin` and the password you found earlier in this course. The URL of the WebGUI user interface is:

https://netcool-evtmanager.apps.labs.ihost.com/ibm/console

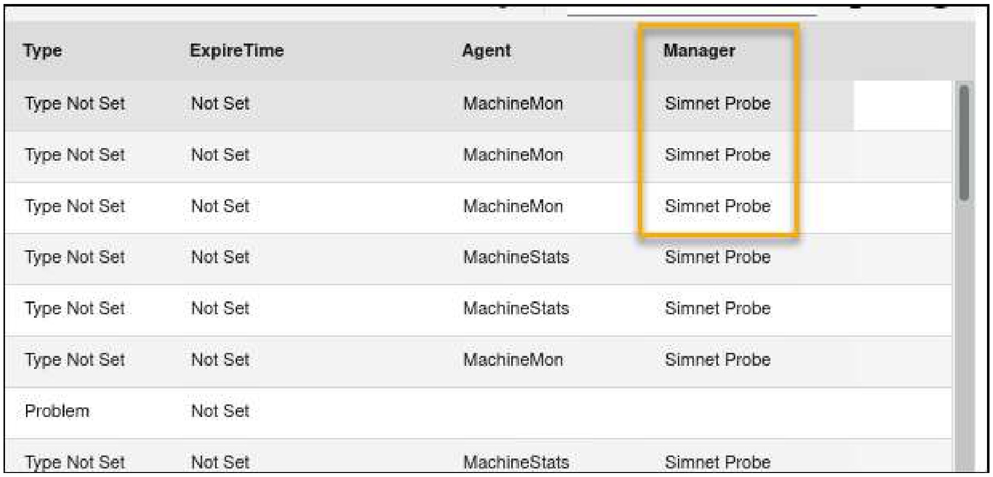
Hint

If you need to find the password for the `icpadmin` user, go back to the lab exercises for Unit 1 to find the password retrieval command.

__ b. If you are prompted with a message about untrusted certificates, click **Close**.

__ c. Click **Incident > Event Viewer** at the top of the page to open an event list.

__ d. Scroll to the right of the event list and find the **Manager** column. The events with `Simnet Probe` in the Manager column came from your probe running on the bastion host. The presence of these events confirms that your on-prem probe is successfully sending events to Event Manager running in Red Hat OpenShift.



Type	ExpireTime	Agent	Manager
Type Not Set	Not Set	MachineMon	Simnet Probe
Type Not Set	Not Set	MachineMon	Simnet Probe
Type Not Set	Not Set	MachineMon	Simnet Probe
Type Not Set	Not Set	MachineStats	Simnet Probe
Type Not Set	Not Set	MachineStats	Simnet Probe
Type Not Set	Not Set	MachineMon	Simnet Probe
Problem	Not Set		
Type Not Set	Not Set	MachineStats	Simnet Probe

__ e. Leave the event list open. You use it again soon.