IBM Tivoli Netcool/OMNIbus Probe for Cloud Monitoring Integration Helm Chart 4.3.0

Reference Guide
March 20, 2020

SC27-9579-03

Note

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 29.

Edition notice

This edition (SC27-9579-03) applies to version 4.3.0 of IBM Tivoli Netcool/OMNIbus Probe for Cloud Monitoring Integration Helm Chart and to all subsequent releases and modifications until otherwise indicated in new editions.

This version replaces SC27-9579-02.

Contents

About this guide

The following sections contain important information about using this guide.

Document control page

Use this information to track changes between versions of this guide.

The Probe for Cloud Monitoring Integration Helm Chart documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/helms/common/Helms.html

Table 1. Document modification history

Document version	Publication date	Comments
SC27-9579-00	June 28, 2019	First IBM publication.
SC27-9579-01	October 31, 2019	Guide updated for version 4.1.0 of the helm chart. The following sections updated: • "Prerequisites" on page 2 • "Secure probe and ObjectServer communication requirements" on page 7 • "Limitations" on page 26
SC27-9579-02	January 31, 2020	Guide updated for version 4.2.0 of the helm chart. "Common Services" on page 3 was added: The following sections updated: • "Prerequisites" on page 2 • "Securing probe and event source communications" on page 11 • "Configurable parameters for the CEM Probe" on page 14 • "Limitations" on page 26
SC27-9579-03	March 20, 2020	Guide updated for version 4.3.0 of the helm chart. The following sections updated: • "Prerequisites" on page 2 • "Integrating Prometheus Alert Manager with Netcool Operations Insight" on page 19 • "Configurable parameters for the CEM Probe" on page 14 • "Limitations" on page 26 • "Troubleshooting" on page 27

Chapter 1. Probe for Cloud Monitoring Integration Helm Chart

The Probe for Cloud Monitoring Integration Helm Chart allows you to deploy the IBM Netcool/OMNIbus Probe for Message Bus onto Kubernetes which starts webhook endpoints to receive notifications in the form of HTTP POST requests from IBM Cloud Event Management (CEM), Logstash and Prometheus Alert Manager.

Note : This Helm Chart is soon to be deprecated. You should use instead, or migrate to, the IBM Netcool Operations Insight Event Integrations Operator when running on Red Hat OpenShift Container Platform. For details see https://www.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/operators/noi_operator/wip/reference/noiop_intro_noi_operator.html. There will be no updates to the deprecated chart.

This guide contains the following sections:

The Knowledge Center contains the following additional topics that contain information that is common to all Helm Charts:

- Specifying the image repository
- Loading PPA packages to IBM Cloud Private
- Exposing the probe service
- Upgrading to a new version of the probe helm charts
- Changing the service type during a helm upgrade

Obtaining the PPA package

You can download the installation package from the **IBM Passport Advantage** website.

Use the **Find by part number** field to search for the following part number: `CC5J0ML`

Chart details

The chart deploys an IBM Netcool/OMNIbus Probe for Message Bus onto Kubernetes which starts webhook endpoints to receive notifications in the form of HTTP POST requests from IBM Cloud Event Management (CEM), Logstash and Prometheus Alert Manager. All probes can be enabled in the same Helm release or enabled individually if necessary. Each probe deployment is fronted by a service.

This chart can be deployed more than once on the same namespace.

Each probe deployment uses a pre-defined probe rules file from a `ConfigMap` to parse the JSON alarms from each event source and maps the attributes to ObjectServer fields. The rules file sets the required Event Grouping field, for example `ScopeID`.

The probe deployments are configured with Horizontal Pod Autoscaler (HPA) to maintain high availability of the service by default. Pod Disruption Budgets (PDB) can be enabled by an Administrator user. HPA and PDB can be customized or disabled to suit your environment.

Prerequisites

This solution requires the following applications:

- IBM Cloud Platform Common Services 3.2.4 must be installed prior to install this chart. See the following topic on the IBM Knowledge Center: Installing Common Services.
- IBM Tivoli Netcool/OMNIbus ObjectServer to be created and running prior to installing the probe either on OpenShift Container Platform 4.3 (OCP) with Common Services or on-premise:
  - For OCP, IBM Netcool Operations Insight 1.6.0.3 is required. See the following topic on the IBM Knowledge Center: Installing Netcool Operations Insight.
  - For on-premise, IBM Tivoli Netcool/OMNIbus 8.1 is required. See the following topic on the IBM Knowledge Center: Creating and running ObjectServers.
- Scope-based Event Grouping automation to be installed, see the following topic on the IBM Knowledge Center: About scope-based event grouping. The probe requires several table fields to be installed in the ObjectServer. For on-premise installation, see Installing scope-based event grouping. The events will be grouped by a preset `ScopeId` in the probe rules file if the event grouping automation triggers are enabled.
- Kubernetes 1.14
- Prometheus 2.3.1 and Prometheus Alert Manager 0.15.0
- IBM Cloud Event Management Helm Chart 2.5.0

Note : Administrator role is a minimum requirement to install this chart.

The chart must be installed by a Administrator to perform the following tasks:

- Enable Pod Disruption Budget policy when installing the chart.
- Perform post-installation tasks such as to configure Prometheus Alert Manager and Logstash in the `kube-system` namespace to add the probe endpoint.
- Retrieve and edit sensitive information from a secret such as the credentials to use to authenticate with the ObjectServer or replace the key database files for secure communications with the ObjectServer.

The chart must be installed by a Cluster Administrator to perform the following tasks in addition to those listed above:

- Obtain the Node IP using kubectl get nodes command if using the NodePort service type.
- Create a new namespace with custom PodSecurityPolicy if necessary.
- Create a service account in the namespace for this chart. Perform one of the following actions:
  - Have the Cluster Administrator pre-create the custom service account in the namespace. This installation requires the service account name to specified to install the chart and can be done by an Administrator.
  - Have the Cluster Administrator perform the installation without specifying a service account name so that the chart generates a service account and use it. When the Helm release is deleted, the service account will also be deleted.
- If secured communication is required or enabled on your Netcool/OMNIbus ObjectService, a per-created secret is required for this chart to establish a secured connection with the ObjectServer.

- Additional ObjectServer fields are required in the `alerts.status` table for IBM CEM integration. For details of the Structured Query Language (SQL) needed to add the required fields, see "Integrating IBM Cloud Event Management (CEM) with Netcool Operations Insight" on page 25
- If you opt to install the chart through command line, then the Helm CLI must be installed. Refer to Installing Helm CLI (helm).

Common Services

If Common Services is not installed, refer to Installing Common Services. Ensure, the following Common Services are enabled prior to install the chart:

```
# ## Common service definition
# management_services:
#   # Default base services
#   tiller: enabled
#   monitoring-crd: enabled
#   mongodb: enabled
#   platform-api: enabled
#   icp-management-ingress: enabled
#   internal-management-ingress: enabled
#   helm-api: enabled
#   helm-repo: enabled
#   mgmt-repo: enabled
#   oidcclient-watcher: enabled
#   secret-watcher: enabled
#   security-onboarding: enabled

#   # Default core services
#   cert-manager: enabled
#   cert-manager-webhook: enabled
#   configmap-watcher: enabled
#   auth-idp: enabled
#   auth-apikeys: enabled
#   auth-pap: enabled
#   auth-pdp: enabled
#   iam-policy-controller: enabled
#   metering: enabled
#   licensing: disabled
#   monitoring: enabled
#   nginx-ingress: enabled
#   catalog-ui: enabled
#   mcm-kui: enabled
#   logging: enabled
#   common-web-ui: enabled

#   # mcm services
#   multicluster-hub: enabled
#   search: enabled
#   key-management: enabled
#   image-security-enforcement: enabled

#   # Deprecated services
#   calico: enabled
#   kube-dns: enabled
```

Resources required

This solution requires the following resources:

- CPU Requested : 100m (100 millicpu)
- Memory Requested : 128Mi (~ 134 MB)

PodSecurityPolicy requirements

This chart requires a PodSecurityPolicy to be bound to the target namespace prior to installation. You can choose either a predefined PodSecurityPolicy or have your cluster administrator create a custom PodSecurityPolicy for you.

The predefined PodSecurityPolicy name `ibm-restricted-psp` has been verified for this chart. If your target namespace is bound to this PodSecurityPolicy, you can proceed to install the chart. The predefined PodSecurityPolicy definitions can be viewed here: https://github.com/IBM/cloud-pak/blob/master/spec/security/psp/README.md

This chart also defines a custom PodSecurityPolicy which can be used to finely control the permissions/capabilities needed to deploy this chart. You can enable this custom PodSecurityPolicy using the ICP user interface or the supplied instructions/scripts in the `pak_extension pre-install` directory. For detailed steps on creating the PodSecurityPolicy see https://www.ibm.com/support/knowledgecenter/SSHTQ/omnibus/helms/all_helms/wip/reference/hlm_common_psp.html

From the user interface, you can copy and paste the following snippets to enable the custom Pod Security Policy

- – Custom PodSecurityPolicy definition:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  annotations:
    kubernetes.io/description: "This policy is the most restrictive,
      requiring pods to run with a non-root UID, and preventing pods from accessing the
host."
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: docker/default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: docker/default
    cloudpak.ibm.com/version: "1.1.0"
  name: ibm-netcool-probe-psp
spec:
  allowPrivilegeEscalation: false
  forbiddenSysctls:
  - '*'
  fsGroup:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  requiredDropCapabilities:
  - ALL
  runAsUser:
    rule: MustRunAsNonRoot
  runAsGroup:
    rule: MustRunAs
    ranges:
    - min: 1
      max: 65535
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  volumes:
  - configMap
  - emptyDir
  - projected
  - secret
  - downwardAPI
  - persistentVolumeClaim
```

– Custom ClusterRole for the custom PodSecurityPolicy:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ibm-netcool-probe-clusterrole
rules:
- apiGroups:
  - policy
  resourceNames:
  - ibm-netcool-probe-psp
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

– RoleBinding for all service accounts in the current namespace. Replace {{ NAMESPACE }} in the template with the actual namespace:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ibm-netcool-probe-rolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ibm-netcool-probe-clusterrole
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts:{{ NAMESPACE }}
```

- From the command line, you can run the setup scripts included under `pak_extensions`.

  As a cluster administrator, the pre-install scripts and instructions are in the following location:

  `pre-install/clusterAdministration/createSecurityClusterPrereqs.sh`

  As team admin/operator the namespace scoped scripts and instructions are in the following location:

  `pre-install/namespaceAdministration/createSecurityNamespacePrereqs.sh`

Security context constraints

The Red Hat OpenShift Container Platform (OCP) provides pod security policies using SecurityContextConstraints (SCC) resources rather than the PodSecurityPolicies (PSP) like all other Kubernetes platforms. SCCs control the actions that a pod can perform and what it has the ability to access. IBM Cloud Private on OCP installations uses SCCs instead of PSPs.

By default, stand-alone OCP uses the SecurityContextConstraints name `restrictedSCC`. However, the predefined SecurityContextConstraints name for this chart is `ibm-restricted-scc`. If your target namespace is bound to this SecurityContextConstraints resource, you can proceed to install the chart.

This chart also defines a custom SecurityContextConstraints resource which can be used to finely control the permissions and capabilities needed to deploy this chart. You can enable this custom resource using the instructions and scripts supplied in the `pak_extension` pre-install directory.

From the user interface, you can copy and paste the following snippets to enable the custom `SecurityContextConstraints`:

- – Custom `SecurityContextConstraints` definition:

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: "This policy is the most restrictive,
      requiring pods to run with a non-root UID, and preventing pods from accessing the
host."
    cloudpak.ibm.com/version: "1.0.0"
  name: ibm-netcool-probe-scc
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegedContainer: false
allowPrivilegeEscalation: false
allowedCapabilities: []
allowedFlexVolumes: []
allowedUnsafeSysctls: []
defaultAddCapabilities: []
defaultPrivilegeEscalation: false
forbiddenSysctls:
  - "*"
fsGroup:
  type: MustRunAs
  ranges:
  - max: 65535
```

```
    min: 1
readOnlyRootFilesystem: false
requiredDropCapabilities:
- ALL
runAsUser:
  type: MustRunAsNonRoot
seccompProfiles:
- docker/default
seLinuxContext:
  type: RunAsAny
supplementalGroups:
  type: MustRunAs
  ranges:
  - max: 65535
    min: 1
volumes:
- configMap
- downwardAPI
- emptyDir
- persistentVolumeClaim
- projected
- secret
```

- From the command line, you can run the setup scripts included under `pak_extensions`.

  As a cluster administrator, the pre-install scripts and instructions are in the following location:

  `pre-install/clusterAdministration/createSecurityClusterPrereqs.sh`

  As team admin/operator the namespace scoped scripts and instructions are in the following location:

  `pre-install/namespaceAdministration/createSecurityNamespacePrereqs.sh`

Creating prerequisite resources for SecurityContextConstraints using the command line

1. Login using `cloudctl` followed by `oc`. In the example below, `MasterNode_IP` refers to the IP address of the master node and `ICP_port` refers to the ICP port.

   ```
   cloudctl login -a https://<MasterNode_IP>:<ICP_port>
   oc login
   ```

2. Extract the pre-installation scripts from the archive under the ibm_cloud_pak/pak_extension directory. Example below shows how to extract the scripts from *ibm-noi-probe-3.10.4.1-x86.tgz* archive.

   ```
   tar xvf ibm-noi-probe-3.10.4.1-x86.tgz ibm_cloud_pak
   ```

3. As a cluster administrator, run the `createSecurityClusterPrereqs.sh` script and provide the target namespace as an argument. This script creates the SecurityContextConstraints and ClusterRole resources. The example command below runs the script on a namespace called `my-probe-namespace`.

   ```
   cd ibm_cloud_pak/pak_extensions
   ./pre-install/clusterAdministration/createSecurityClusterPrereqs.sh my-probe-namespace
   ```

4. As an administrator or cluster administrator, run the namespace scoped `createSecurityNamespacePrereqs.sh` script and provide the target namespace as an argument. This script creates the RoleBinding resource for service accounts in the target namespace. The namespace must be created prior to running this script. The example command below runs the script on a namespace called `my-probe-namespace`.

   ```
   cd ibm_cloud_pak/pak_extensions
   ./pre-install/namespaceAdministration/createSecurityNamespacePrereqs.sh my-probe-namespace
   ```

5. Verify that the custom SCC has been created.

   ```
   oc get scc | grep <SCC name>
   ```

6. You can now proceed to install the chart in the namespace with the custom SecurityContextConstraints.

Secure probe and ObjectServer communication requirements

There are several mechanisms to secure Netcool/OMNIbus. Authentication can be used to restrict user access while Secure Sockets Layer (SSL) protocol can be used for different levels of encryption.

The probe connection mode is dependent on the server component configuration. Check with your Netcool/OMNIbus Administrator whether the server is configured with either secured mode enabled without SSL, SSL enabled with secured mode disabled, or secured mode enabled with SSL protected communications

For more details on running the ObjectServer in secured mode, refer to Running the ObjectServer in secure mode on the IBM Knowledge Center. For more details on SSL protected communications, refer to Using SSL for client and server communications on the IBM Knowledge Center.

The chart must be configured according to the server components setup in order to establish a secured connection with or without SSL. This can be configured by setting the `netcool.connectionMode` chart parameter with one of the following options:

- `Default` - This is the default mode. Use this mode to connect with the ObjectServer with neither secure mode nor SSL.
- `AuthOnly` - Use this mode when the ObjectServer is configured to run in secured mode without SSL.
- `SSLOnly` - Use this mode when the ObjectServer is configured with SSL without secure mode.
- `SSLAndAuth` - Use this mode the ObjectServer is configured with SSL and secure mode.

To secure the communications between probe clients and the ObjectServer, you must perform the following tasks that must be completed before installing the chart:

1. “Determine files required for the secret” on page 7
2. “Preparing credential files for authentication” on page 8
3. “Preparing the key database file for SSL communication” on page 9
4. “Create the probe-server communication secret” on page 10

If you are using the `Default` mode, you can skip these steps and proceed configuring the chart with your ObjectServer connection details.

Note : There are several known limitations listed in the Limitations section when securing probe communications.

Determine files required for the secret

For `AuthOnly`, `SSLOnly` and `SSLAndAuth` options, a secret is required to support the secured connection prior installing the chart. The secret name should be set in the `netcool.secretName` parameter when configuring the chart or left unset if you want to use the default connection mode.

Several fields are required in the Secret depending on the connection mode that will be configured in the chart.

For secured mode (`AuthOnly`), the following fields are required:

- `encryption.keyfile` - An encryption key file to decrypt your password.
- `AuthUserName` - Specify the username to authenticate with the ObjectServer.
- `AuthPassword` - Specify the password for the specified user. It is recommended to encrypt this string using the encryption key file above.

For SSL enabled mode (`SSLOnly`), the following fields are required:

- `omni.kdb` - A Key Database file. This Key Database file should contain the root Certificate Authority certificate used by the server components.
- `omni.sth` - The Key Database Stash file. This file is required for automatic authentication.

For secured and SSL enabled mode (`SSLAndAuth`), the following fields are required:

- `encryption.keyfile` - An encryption key file to decrypt your password.
- `AuthUserName` - Specify the username to authenticate with the ObjectServer.
- `AuthPassword` - Specify the password of the specified user. It is recommended to encrypt this string using the encryption key file above.
- `omni.kdb` - A Key Database file. This Key Database file should contain the root Certificate Authority certificate used by the server components.
- `omni.sth` - The Key Database Stash file. This file is required for automatic authentication.

You only need to create a single secret with the required fields which can be used by both the Logstash and Prometheus probes.

Preparing credential files for authentication

For the probe to connect with a secured ObjectServer, the probe must be configured with a valid credential otherwise the ObjectServer will reject the connection. The credentials will be set in a secret for the pods to retrieve the secret and configure the probe. Follow the steps below to prepare the required files prior creating the secret.

If you are connecting to an ObjectServer on IBM Cloud Private, you need to extract the CA certificate from the proxy deployed with the ibm-netcool-prod deployment and use the proxy to secure the connection using TLS. For more information on how to get the connection details and the proxy certificate, see Connecting event sources.

Note :

This section is only applicable if you are using either the `AuthOnly` or SSLAndAuth connection mode.

Some of steps below use utility commands in the probe image. Update the image repository in the each command to pull the image from IBM Cloud Private's private registry or pull the image to your local file system first. For more details about how to pull the image to your local file system, see Pushing and pulling images.

1. Contact your Netcool/OMNIbus Administrator to obtain the credentials that should be used by probe clients to authenticate with the ObjectServer. Optionally, you can also request an encrypted password with its encryption key file from the administrator to be used in the secret.
2. Create a temporary workspace directory if necessary. The following steps uses `/tmp/probe` as the workspace directory.

   ```
   # Create a temporary directory to mount to the container
   $ mkdir /tmp/probe
   ```

3. Create a new file called `username.txt` and insert the username string into the file. For example:

   ```
   echo -n "probe_user" > /tmp/probe/username.txt
   ```

4. If an encryption key file is not provided by the ObjectServer administrator, you (administrator) need to create a new encryption key file to encrypt the password. Otherwise, you can skip this step. To create an encryption file using the `nco_keygen` command from the probe Docker image, use the following commands:

   ```
   # Start the probe container and create an encryption key file in the mounted directory.
   # Use valid 256 for the key length.
   $ docker run -e LICENSE=accept \
   --entrypoint /bin/bash -it \
   -v /tmp/probe:/home/netcool netcool-probe-messagebus:12.0.0-amd64 \
   -c "/opt/IBM/tivoli/netcool/omnibus/bin/nco_keygen \
   -o /home/netcool/encryption.keyfile \
   -l 256"
   ```

5. Encrypt the password string with the encryption key file using the `nco_aes_crypt` tool with AES_FIPS cipher and output the encrypted password into a file called `password.txt` in the mounted

directory. Example command to encrypt a password (AVeryStrongPassw0rd) is shown below. Replace the password string in the command with your own.

```
# Encrypt the password
$ docker run -e LICENSE=accept \
--entrypoint /bin/bash -it \
-v /tmp/probe:/home/netcool \
netcool-probe-messagebus:12.0.0-amd64 \
-c "/opt/IBM/tivoli/netcool/omnibus/bin/nco_aes_crypt \
-c AES_FIPS \
-k /home/netcool/encryption.keyfile AVeryStrongPassw0rd \
-o /home/netcool/password.txt"
```

6. Verify that the following files are created in the temporary workspace directory:
   - encryption.keyfile
   - password.txt
   - username.txt

```
$ ls -1 /tmp/probe/
encryption.keyfile
password.txt
username.txt
```

Preparing the key database file for SSL communication

For the probe to connect using SSL, the probe needs the server certificate in its key database in order to authenticate with the server when establishing a connection. The key database files are then put in a secret. Follow the steps below to set up the key database files and add the server certificate.

Note :

This is section is only applicable if you are using either the SSLOnly or SSLAndAuth connection.

Some of steps below use utility commands in the probe image. Update the image repository in the each command to pull the image from IBM Cloud Private's private registry or pull the image to your local file system first. For more details about how to pull the image to your local file system, see Pushing and pulling images.

1. Contact your Netcool/OMNIbus Administrator to extract the SSL certificate from the server key database which will then be added into the key database of the probe client, see Extracting certificates from a key database. Use the nc_gskcmd to extract the certificate from the server's key database on the server host, for example:

```
$NCHOME/bin/nc_gskcmd -cert -extract \
-db "$NCHOME/etc/security/keys/omni.kdb" \
-pw password \
-label "keylabel" \
-target "$NCHOME/etc/security/keys/certname.arm"
```

   Where *password* is the password for the key database, *keylabel* is the description of the certificate in the key database (specify this value as a quoted string), and *certname* is the name of the certificate that you want to extract. Specify the path to the certificate as a quoted string. A sample command to extract the a root CA certificate with the label "NCOMS_CA" label into the file ncoms-ca-cert.arm:

```
$NCHOME/bin/nc_gskcmd -cert -extract \
-db "$NCHOME/etc/security/keys/omni.kdb" \
-pw password \
-label "NCOMS_CA" \
-target "$NCHOME/etc/security/keys/ncoms-ca-cert.arm"
```

2. Create a temporary workspace directory if necessary. The following steps use /tmp/probe as the workspace directory.

```
$ mkdir /tmp/probe
```

3. Copy the server certificate into the workspace directory.

```
$ cp ncoms-ca-cert.arm /tmp/probe
```

4. Create a key database file with a strong password which meets the password requirements, see Creating a key database using nc_gskcmd.

   For example, to create a key database file that is valid for 366 days:

```
# Create a Key Database file
$ docker run -e LICENSE=accept \
--entrypoint /bin/bash -it \
-v /tmp/probe:/home/netcool \
netcool-probe-messagebus:12.0.0-amd64 \
-c "/opt/IBM/tivoli/netcool/bin/nc_gskcmd -keydb -create \
-db \"/home/netcool/omni.kdb\" \
-pw AVeryStrongKeyDBPasSw0rd \
-stash \
-expire 366"
```

5. Add the server certificate into the key database file. The following example command adds the server certificate into the `ncoms-ca-cert.arm` file into the key database with the label `NCOMS_CA`:

```
# Add server root CA into Key DB
$ docker run -e LICENSE=accept \
--entrypoint /bin/bash -it \
-v /tmp/probe:/home/netcool \
netcool-probe-messagebus:12.0.0-amd64 \
-c "/opt/IBM/tivoli/netcool/bin/nc_gskcmd -cert -add \
-db \"/home/netcool/omni.kdb\" \
-pw AVeryStrongKeyDBPasSw0rd \
-label \"NCOMS_CA\" \
-file \"/home/netcool/ncoms-ca-cert.arm\""
```

6. Optionally, you can verify that the server certificate has been added into the key database using the following command:

```
# Add server root CA into Key DB
$ docker run -e LICENSE=accept \
--entrypoint /bin/bash -it \
-v /tmp/probe:/home/netcool \
netcool-probe-messagebus:12.0.0-amd64 \
-c "/opt/IBM/tivoli/netcool/bin/nc_gskcmd -cert -list \
-db \"/home/netcool/omni.kdb\" \
-pw AVeryStrongKeyDBPasSw0rd"
Certificates found
default, - personal, ! trusted, # secret key
- NCOMS_CA
```

Create the probe-server communication secret

After the preparing the required files, use the `kubectl create secret generic` command with the `--from-file` option to create a Kubernetes secret.

Depending on the connection mode that is required by the ObjectServer, create a Secret with the required fields. The secret name should contain the helm release name as a prefix for easy reference.

Example commands are shown below for each connection mode using `my-probe` as the release name to create a secret called `my-probe-comms` in the `icp-noi` namespace.

- To create a secret for secured communication (`AuthOnly` connection mode), use:

```
kubectl create secret generic my-probe-comms \
--namespace icp-noi \
--from-file=encryption.keyfile=/tmp/probe/encryption.keyfile \
--from-file=AuthUserName=/tmp/probe/username.txt \
--from-file=AuthPassword=/tmp/probe/password.txt
```

- To create a secret for SSL enabled communication (`SSLOnly` connection mode), use:

```
kubectl create secret generic my-probe-comms \
--namespace icp-noi \
```

```
--from-file=omni.kdb=/tmp/probe/omni.kdb \
--from-file=omni.sth=/tmp/probe/omni.sth
```

- To create a secret for SSL enabled and secured communication (SSLAndAuth connection mode), use:

```
kubectl create secret generic my-probe-comms \
--namespace icp-noi \
--from-file=encryption.keyfile=/tmp/probe/encryption.keyfile \
--from-file=AuthUserName=/tmp/probe/username.txt \
--from-file=AuthPassword=/tmp/probe/password.txt \
--from-file=omni.kdb=/tmp/probe/omni.kdb \
--from-file=omni.sth=/tmp/probe/omni.sth
```

Note : Each field key must follow the expected names `AuthUserName`, `AuthPassword`, `encryption.keyfile`, `omni.kdb` and `omni.sth` and is case sensitive. If any of the field key is misspelled, the Pod might fail to mount to the secret to obtain the secret.

Tip : If you have a different encryption key file name, you can map the name of the file to `encryption.keyfile` when creating the secret as shown below. The sample command below renames `ProbeEncryption.keyfile` to the expected field name.

```
kubectl create secret generic my-probe-comms \
--namespace icp-noi \
--from-file=encryption.keyfile=/tmp/probe/ProbeEncryption.keyfile  \
--from-file=AuthUserName=/tmp/probe/username.txt \
--from-file=AuthPassword=/tmp/probe/password.txt \
--from-file=omni.kdb=/tmp/probe/omni.kdb \
--from-file=omni.sth=/tmp/probe/omni.sth
```

Remember to clean up this directory after you have successfully created the Kubernetes secret.

Securing probe and event source communications

To secure the communications between the probe and event sources such as Prometheus, Logstash or IBM CEM in the same cluster, you can enable Mutual Transport Layer Security (mTLS) in Service Mesh to encrypt cluster data network traffic. For more information, refer to the ServiceMeshControlPlane parameters section in the Service Mesh installation, usage, and release notes.

Role-based access control

Role-Based Access Control (RBAC) is applied to the chart by using a custom service account having a specific role binding. RBAC provides greater security by ensuring that the chart operates within the specified scope.

Currently, RBAC is only applicable to the IBM Netcool/OMNIbus Probe Cloud Monitoring Integration chart.

The following are the options to apply RBAC to the chart:

- "Deploying the chart with pre-created role and role binding resources" on page 11: The Cluster Administrator per-creates the role and role binding resources prior to deploying the chart. With the role binding resources in place, the chart may be assigned to another user such as an Administrator to perform the deployment. This option may require additional setup prior to setting up the chart, but it enables the RBAC resources to be easily applied to multiple chart deployments in the namespace.
- "Deploying the chart with the role and role binding resources included with the chart" on page 13: The Cluster Administrator deploys the chart with the role and role binding resources being included with the chart. This option automates the creation of the RBAC resources for the chart deployment, but the chart deployment must be performed by the Cluster Administrator. The RBAC resources is managed by the chart and will be removed when the release is deleted.

Deploying the chart with pre-created role and role binding resources

The Cluster Administrator will apply these steps to pre-create the required RBAC resources, namely: service account, role and role bindings, before the chart is deployed. Once the RBAC resources are created, the chart deployment may be done by another user eg. Administrator.

1. As the Cluster Administrator, create a YAML file called `serviceAccount.yaml` with the contents below, where `Release_Name` is the release name of the chart, `Namespace` is the target namespace, `Probe_Name` is the probe name and `Secret_Name` is the image pull Secret.

```
apiVersion: v1
kind: ServiceAccount
metadata:
    name: <Release_Name>-ibm-netcool-probe-sa
    namespace: <Namespace>
    labels:
        app.kubernetes.io/name: "<Probe_Name>"
        helm.sh/chart: "<Probe_Name>"
        app.kubernetes.io/instance: "<Release_Name>"
        release: "<Release_Name>"
        app.kubernetes.io/component: "common"
imagePullSecrets:
- name: <Secret_Name>
```

   Create the Service Account using the following command:

```
kubectl apply -f serviceAccount.yaml
```

2. As the Cluster Administrator, create a YAML file called `role.yaml` with the contents below, where `Release_Name` is the release name of the chart and `Namespace` is the target namespace.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
    name: <Release_Name>-ibm-netcool-probe-role
    namespace: <Namespace>
    labels:
        app.kubernetes.io/name: "<Probe_Name>"
        helm.sh/chart: "<Probe_Name>"
        app.kubernetes.io/instance: "<Release_Name>"
        release: "<Release_Name>"
        app.kubernetes.io/component: "common"
rules:
- apiGroups: [""] # "" indicates the core API group
    resources: ["secrets"]
    verbs: ["get", "list"]
```

   Create the Role using the following command:

```
kubectl apply -f role.yaml
```

3. As the Cluster Administrator, create a YAML file called `roleBinding.yaml` with the contents below, where `Release_Name` is the release name of the chart and `Namespace` is the target namespace.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
    name: <Release_Name>-ibm-netcool-probe-rolebinding
    namespace: <Namespace>
    labels:
        app.kubernetes.io/name: "<Probe_Name>"
        helm.sh/chart: "<Probe_Name>"
        app.kubernetes.io/instance: "<Release_Name>"
        release: "<Release_Name>"
        app.kubernetes.io/component: "common"
subjects:
- kind: ServiceAccount
    name: <Release_Name>-ibm-netcool-probe-sa
namespace: <Namespace>
roleRef:
    kind: Role
    name: <Release_Name>-ibm-netcool-probe-role
    apiGroup: rbac.authorization.k8s.io
```

   Create the Role Binding using the following command:

```
kubectl apply -f roleBinding.yaml
```

4. Set the service account name created earlier in the chart's **global.serviceAccountName** parameter. As an Administrator, edit `values.yaml` and update the `serviceAccountName` with the name of the Service Account created earlier.

```
global:
    image:
        secretName: "<Secret_Name>"
    serviceAccountName: "<Release_Name>-ibm-netcool-probe-sa"
```

5. Perform all required edits to `values.yaml` and deploy the chart.

Deploying the chart with the role and role binding resources included with the chart

When the Cluster Administrator applies these steps, the required RBAC resources ie. service account, role and role bindings will be created when the chart is deployed. It should be noted that the same RBAC resources will be removed automatically when the Helm release is deleted.

1. As the Cluster Administrator, edit `values.yaml` and specify the image pull Secret for the `secretName` parameter. Update the `serviceAccountName` parameter to be blank. These settings enable the chart to automatically create the require service account, role and role bindings using the specified Secret to pull the image.

```
global:
    image:
        secretName: "<Secret_Name>"
    serviceAccountName: ""
```

2. Perform all required edits to `values.yaml` and deploy the chart.

Installing the chart

To install the chart, use the following steps:

1. Extract the helm chart archive and customize `values.yaml`. The configuration section lists the parameters that can be configured during installation.
2. Install the chart with the release name `my-probe` using the configuration specified in the customized values.yaml using following command:

   `helm install --tls --namespace <your pre-created namespace> --name my-probe -f values.yaml stable/ibm-netcool-probe`

   Where: *my-probe* is the release name for the chart.

   Helm searches for the `ibm-netcool-probe` chart in the helm repository called `stable`. This assumes that the chart exists in the stable repository.

   Tip : You can list all releases using `helm list --tls` or you can search for a chart using **helm search**.

   The command deploys on the Kubernetes cluster using a default configuration. For a list of the parameters that you can configure during installation see "Configurable parameters for the CEM Probe" on page 14.

Verifying the chart

See the instructions at the end of the helm installation for chart verification. The instructions can also be displayed by viewing the installed helm release under **Menu -> Workloads -> Helm Releases** or by running the following command:

`helm status <release> --tls`

Uninstalling the chart

To uninstall the chart, use the following command:

```
$ helm delete my-probe --purge --tls
```

Where: *my-probe* is the release name for the chart.

The command removes all the Kubernetes components associated with the chart and deletes the release.

Clean up any prerequisites that were created

As a Cluster Administrator, run the cluster administration cleanup script included under `pak_extensions` to clean up cluster scoped resources when appropriate. The following example command runs the script on a namespace called `my-probe-namespace`:

```
post-delete/clusterAdministration/deleteSecurityClusterPrereqs.sh my-probe-namespace
```

As a Cluster Administrator, run the namespace administration cleanup script included under `pak_extensions` to clean up namespace scoped resources when appropriate. The following example command runs the script on a namespace called `my-probe-namespace`:

```
post-delete/namespaceAdministration/deleteSecurityNamespacePrereqs.sh my-probe-namespace
```

Configuring the chart

The integration requires configuration of the following components:

- This chart (to deploy the Netcool/OMNIbus probes).
- Prometheus Alert Manager (to add a new `receiver` to direct notification to the probe and to apply Prometheus alert rules).
- Logstash pipeline (to add an `http output` to send notification to the probe).

Note : If you have IBM CEM installed, see Integrating IBM Cloud Event Management (CEM) with Netcool Operations Insight.

The following topics describe how to configure the integration.

Configurable parameters for the CEM Probe

You use parameters to specify how the probe interacts with the device. You can override the chart's default parameter settings during installation.

The following table describes the configurable parameters for this chart and lists their default values.

Configurable parameters

Parameter name	Description
license	The license state of the image being deployed. Enter `accept` to install and use the image. The default value is `not accepted`.
image.repository	Probe image repository. Update this repository name to pull from a private image repository. For details see Specifying the image repository. The default value is `netcool-probe-messagebus`, and must not be changed.

Parameter name	Description
image.tag	The image tag. The default value is `12.0.0-amd64`.
image.testRepository	Utility image repository. Update this repository name to pull from a private image repository. The default is `netcool-integration-util`, and must not be changed.
image.testImageTag	Utility image tag. The default is `3.1.0-amd64`.
image.pullPolicy	The image pull policy. The default value is `Always`.
global.image.secretName	The name of the secret containing the docker config to pull the image from a private repository. Leave this parameter blank if the probe image already exists in the local image repository or the Service Account has a been assigned with an Image Pull Secret. The default value is `nil`.
global.serviceAccountName	Description: Name of the service account to be used by the helm chart. If the Cluster Administrator has already created a service account in the namespace, specify the name of the service account here. If left blank, the chart will automatically create a new service account in the namespace when it is deployed. This new service account will be removed from the namespace when the chart is removed. The default is `nil`.
netcool.connectionMode	The connection mode to use when connecting to the Netcool/OMNIbus ObjectServer. Refer to Securing probe and ObjectServer communications for more details. **Note:** Refer to the limitations section for more details on available connection modes for your environment. The default is `default`.
netcool.primaryServer	The primary Netcool/OMNIbus server to connect to. This is usually set to `NCOMS` or `AGG_P`. The default value is `nil`.
netcool.primaryHost	The host of the primary Netcool/OMNIbus server. Specify the ObjectServer hostname or IP address. The default value is `nil`
netcool.primaryPort	The port of the primary Netcool/OMNIbus server. The default value is `nil`.
netcool.backupServer	The backup Netcool/OMNIbus server to connect to. If the **backupServer**, **backupHost** and **backupPort** parameters are defined in addition to the **primaryServer**, **primaryHost**, and **primaryPort** parameters, the probe will be configured to connect to a virtual object server pair called `` `AGG_V` ``. If no backup ObjectServer is configured, only the primary server parameters will be used. The default value is `nil`.

Parameter name	Description
netcool.backupHost	The host of the backup Netcool/OMNIbus server. Specify the ObjectServer hostname or IP address. The default value is `nil`
netcool.backupPort	The port of the backup Netcool/OMNIbus server. The default value is `nil`.
netcool.secretName	This is a pre-created secret for `AuthOnly`, `SSLOnly` or `SSLAndAuth` connection mode. Certain fields are required depending on the connection modes. The default is `nil`.
logstashProbe.enabled	Set this parameter to `true` to enable the probe for Logstash. The default value is `false`.
logstashProbe.replicaCount	The number of deployment replicas of the Logstash probe. This will be ignored if `logstashProbe.autoscaling.enabled=true` and will use **minReplicas** as the **replicaCount**. The default value is 5.
logstashProbe.service.type	The Logstash probe service type. Valid options are: `NodePort` or `ClusterIP`. The default value is `ClusterIP`.
logstashProbe.service.externalPort	The external port that the Logstash probe is running on. The default value is `80`.
logstashProbe.ingress.enabled	Set this parameter to `true` to enable Ingress. Use this parameter to create an Ingress record for the Logstash probe. **Note:** This should be used with **service.type**: `ClusterIP`. The default value is `false`.
logstashProbe.ingress.hosts	This parameter sets the virtual host names for the same IP address. The Helm release name will be appended as a prefix. The default value is `netcool-probe-logstash.local`.
logstashProbe.autoscaling.enabled	Set this parameter to `false` to disable auto-scaling. The default value is `true`.
logstashProbe.autoscaling.minReplicas	The minimum number of probe replicas. The default value is 2.
logstashProbe.autoscaling.maxReplicas	The maximum number of probe replicas. The default value is 6.
logstashProbe.autoscaling.cpuUtil	The target percentage CPU utilization. For example, enter `60` for 60% target utilization. The default value is `60`.
logstashProbe.poddisruptionbudget.enabled	Set this parameter to `true` to enable Pod Disruption Budget to maintain high availability during node maintenance. Administrator role or higher is required to enable Pod Disruption Budget on clusters with Role Based Access Control. The default value is `false`.

Parameter name	Description
logstashProbe.poddisruptionbudget.minAvailable	The minimum number of available pods during node drain. This can be set to a number or a percentage, for example: 1 or 10%. **Caution:** Setting this parameter to 100%, or to the number of replicas, may block node drains entirely. The default value is 1.
prometheusProbe.enabled	Set this parameter to `true` to enable the probe for Prometheus. The default value is `true`.
prometheusProbe.replicaCount	The number of deployment replicas of the Prometheus Probe. This will be ignored if `prometheusProbe.autoscaling.enabled=true` and will use the **minReplicas** as the **replicaCount**. The default value is 1.
prometheusProbe.service.type	The Prometheus probe service type. Valid options are `NodePort` or `ClusterIP`. The default value is `ClusterIP`.
prometheusProbe.service.externalPort	The external port that the Prometheus probe is running on. The default value is 80.
prometheusProbe.ingress.enabled	Set this parameter to `true` to enable Ingress. Use this parameter to create an Ingress record for the Prometheus probe. **Note:** This should be used with service.type: ClusterIP. The default value is `false`.
prometheusProbe.ingress.hosts	This parameter sets the virtual host names for the same IP address. The Helm release name will be appended as a prefix. The default value is `netcool-probe-prometheus.local`.
prometheusProbe.autoscaling.enabled	Set this parameter to `false` to disable auto-scaling. The default value is `true`.
prometheusProbe.autoscaling.minReplicas	The minimum number of probe replicas. The default value is 1.
prometheusProbe.autoscaling.maxReplicas	The maximum number of probe replicas. The default value is 3.
prometheusProbe.autoscaling.cpuUtil	The target percentage CPU utilization. For example, enter 60 for 60% target utilization. The default value is 60.
prometheusProbe.poddisruptionbudget.enabled	Set this parameter to `true` to enable Pod Disruption Budget to maintain high availability during a node maintenance. Administrator role or higher is required to enable Pod Disruption Budget on clusters with Role Based Access Control. The default value is `false`.
prometheusProbe.poddisruptionbudget.minAvailable	The minimum number of available pods during node drain. This can be set to a number or a percentage, for example: 1 or 10%. **Caution:** Setting this parameter to 100%, or to the number of replicas, may block node drains entirely. The default value is 1.

Parameter name	Description
cemProbe.enabled	Set this parameter to `true` to enable the probe for CEM. The default value is `true`.
cemProbe.tlsEnabled	Enables the CEM TLS Webhook for secure connectivity between the probe and CEM. Default is `false`.
cemProbe.replicaCount	The number of deployment replicas of the CEM Probe. This will be ignored if `cemProbe.autoscaling.enabled=true` and will use the **minReplicas** as the **replicaCount**. The default value is `1`.
cemProbe.service.type	The CEM Probe service type. Valid options are `NodePort` or `ClusterIP`. The default value is `ClusterIP`.
cemProbe.service.externalPort	The external port that the CEM Probe is running on. The default value is `80`.
cemProbe.autoscaling.enabled	Set this parameter to `false` to disable auto-scaling. The default value is `true`.
cemProbe.autoscaling.minReplicas	The minimum number of probe replicas. The default value is `1`.
cemProbe.autoscaling.maxReplicas	The maximum number of probe replicas. The default value is `3`.
cemProbe.autoscaling.cpuUtil	The target percentage CPU utilization. For example, enter `60` for `60%` target utilization. The default value is `60`.
cemProbe.poddisruptionbudget.enabled	Set this parameter to `true` to enable Pod Disruption Budget to maintain high availability during a node maintenance. Administrator role or higher is required to enable Pod Disruption Budget on clusters with Role Based Access Control. The default value is `false`.
cem.poddisruptionbudget.minAvailable	The minimum number of available pods during node drain. This can be set to a number or a percentage, for example: `1` or `10%`. **Caution:** Setting this parameter to `100%`, or to the number of replicas, may block node drains entirely. The default value is `1`.
probe.messageLevel	The probe log message level. The default value is `warn`.

Parameter name	Description
probe.sslServerCommonName	Set this parameter to a comma-separated list of acceptable SSL common names for when connecting to the ObjectServer using SSL. This should be set when the `CommonName` field of the received certificate does not match the name specified by the **`primaryServer`** property. When a **`backupServer`** is specified, the probe creates an ObjectServer pair with the name `AGG_V`. Set this parameter if the `CommonName` of the certificate does not match `AGG_V`. The default is `nil`.
resources.limits.cpu	The container CPU limit. The default value is `500m`.
resources.limits.memory	The container memory limit. The default value is `512Mi`.
resources.requests.cpu	The container CPU requested. The default value is `100m`.
resources.requests.memory	The container memory requested. The default value is `128Mi`.
arch	The worker node architecture. This is set to `amd64`, and cannot be changed.

Integrating Prometheus Alert Manager with Netcool Operations Insight

Getting the Probe Webhook Endpoint URL

To modify the default Prometheus configuration, use the following steps:

1. After a successful deployment, get the probe's `Endpoint Host` and `Port` from the **Workloads > Deployments** page.
   - If **`prometheusPobe.service.type`** is set to `ClusterIP`, the full webhook URL will have the following format:

     ```
     http://<service name>.<namespace>:<externalPort>/probe/webhook/prometheus
     ```

     To obtain the service name and port using the command line, use the following commands substituting `<namespace>` with the namespace where the release is deployed and `<release_name>` with the Helm release name.

     ```
     # Get the Service name

     export SVC_NAME=$(kubectl get services --namespace <namespace> -l
     "app.kubernetes.io/instance=<release_name>,app.kubernetes.io/
     component=prometheusprobe" -o jsonpath="{.items[0].metadata.name}")

     # Get the Service port number

     export SVC_PORT=$(kubectl get services --namespace <namespace> -l
     "app.kubernetes.io/instance=<release_name>,app.kubernetes.io/
     component=prometheusprobe" -o jsonpath="{.items[0].spec.ports[0].port}")
     ```
   - If **`prometheusPobe.service.type`** is set to `Nodeport`, the full webhook URL will have the following format:

     ```
     http://<External IP>:<Node Port>/probe/webhook/prometheus
     ```

To obtain the NodePort number using the command line, use the following commands substituting <namespace> with the namespace where the release is deployed and <release_name> with the Helm release name.

```
# Get the NodePort number from the Service resource

export NODE_PORT_PROMETHEUS=$(kubectl get services --namespace <namespace>
-l "app.kubernetes.io/instance=<release_name>,app.kubernetes.io/
component=prometheusprobe" -o
jsonpath="{.items[0].spec.ports[0].nodePort}")

# On OCP 4.2 or later with IBM Cloud Pak Common Services, you can obtain
the External IP from the IBM Cloud Cluster Info Configmap using the command
below:

export NODE_IP_PROMETHEUS=$(kubectl get configmap --namespace kube-public
ibmcloud-cluster-info -o jsonpath="{.data.proxy_address}")

echo http://$NODE_IP_PROMETHEUS:$NODE_PORT_PROMETHEUS/probe/webhook/
prometheus
```

Modifying Prometheus Alert Manager and Alert Rules Configuration for OCP Monitoring

1. Determine the Prometheus Alert Manager configuration secret in the cluster. The default Secret that contains the Alert Manager configuration is in the `openshift-monitoring`. Refer to Applying custom Alertmanager configuration.
2. A sample Alert Manager configuration with the probe webhook config applied is shown below. The sample endpoint `http://my-probe-ibm-netcool-probe-prometheusprobe.default.svc:80/probe/webhook/prometheus` is used where `my-probe-ibm-netcool-probe-prometheusprobe.default.svc` is the probe's service name in the `default` namespace.

```
global:
  resolve_timeout: '5m'
receivers:
- name: 'null'
- name: 'netcool_probe'
  webhook_configs:
  - url: 'http://my-probe-ibm-netcool-probe-prometheusprobe.default.svc:80/probe/webhook/
prometheus'
    send_resolved: true
route:
  group_by:
  - alertname
  group_interval: 5m
  group_wait: 30s
  receiver: netcool_probe
  repeat_interval: 5s
  routes:
  - receiver: netcool_probe
    match:
      alertname: Watchdog
```

   Note : The **`send_resolved`** flag should be set to `true` so that the probe receives resolution events.
3. Apply the updated Alert Manager configuration file.
4. You can apply custom alerting rules by referring to Managing cluster alerts
5. Verify that your probe is receiving the OCP Monitoring alerts and events appear on the Netcool/OMNIbus Event List.

Modifying Prometheus Alert Manager and Alert Rules Configuration for IBM Cloud Platform Common Services Monitoring

1. Determine the Prometheus Alert Manager config map in the cluster. In this procedure, the config map in the `kube-system` namespace are `monitoring-prometheus-alertmanager`. The following steps use this config maps as an example.
2. Edit the Prometheus Alert Manager ConfigMap to add a new receiver in the `receivers` section. If a separate Prometheus is deployed, determine the Alert Manager ConfigMap and add the new receiver. To do this using the command line, load the `monitoring-prometheus-alertmanager` ConfigMap into a file using the following command:

   ```
   kubectl get configmap monitoring-prometheus-alertmanager --namespace=kube-system -o yaml > alertmanager.yaml
   ```

3. Edit the `alertmanager.yaml` file and add a new webhook receiver configuration. A sample configmap configuration is shown below. Use the full webhook URL from Step 2 in the **`url`** parameter.

   ```
   $ cat alertmanager.yaml
   apiVersion: v1
   data:
     alertmanager.yml: |-
       global:
       receivers:
       - name: 'netcool_probe'
         webhook_configs:
         - url: 'http://<ip_address>:<port>/probe/webhook/prometheus'
           send_resolved: true

       route:
         group_wait: 10s
         group_interval: 5m
         receiver: 'netcool_probe'
         repeat_interval: 3h
   kind: ConfigMap
   metadata:
     creationTimestamp: 2018-04-18T02:38:14Z
     labels:
       app: monitoring-prometheus
       chart: ibm-icpmonitoring-1.3.0
       component: alertmanager
       heritage: Tiller
       release: monitoring
     name: monitoring-prometheus-alertmanager
     namespace: kube-system
     resourceVersion: "1856489"
     selfLink: /api/v1/namespaces/kube-system/configmaps/monitoring-prometheus-alertmanager
     uid: 8aef5f39-42b1-11e8-bd3d-0050569b6c73
   ```

   Note : The **`send_resolved`** flag should be set to `true` so that the probe receives resolution events.

4. Save the changes in the file and replace the ConfigMap using the following command:

   ```
   $ kubectl replace configmap monitoring-prometheus-alertmanager --namespace=kube-system -f alertmanager.yaml

   configmap "monitoring-prometheus-alertmanager" replaced
   ```

5. Review the sample alert-rules CRD YAML below. You may update the rules or add more rules to generate more alerts to monitor your cluster.

   The Message Bus Probe rules file expects the following attributes from the alerts generated by Prometheus Alert Manager:

   `labels.severity`: The severity of the alert. This should be set to `critical`, `major`, `minor`, or `warning`. This is mapped to the `Severity` field in the ObjectServer `alerts.status` table.

   `labels.instance`: The instance generating the alert. This is mapped to the `Node` field in the ObjectServer `alerts.status` table.

   `labels.alertname`: The alert rule name. This is mapped to the `AlertGroup` field in the ObjectServer `alerts.status` table.

`annotations.description`: (Optional) The full description of the alert. This is mapped to the `Summary` field in the ObjectServer `alerts.status` table.

`annotations.summary`: A short description or summary of the alert. This is mapped to the `Summary` field in the ObjectServer `alerts.status` table if `annotations.description` is unset.

`annotations.type`: The alert type. For example, `Container`, `Service`, or `Service`. This is mapped to the `AlertKey` field in the ObjectServer `alerts.status` table.

`labels.release`: (Optional) If set, will be mapped to the `ScopeId` field in the ObjectServer `alerts.status` table which will be used as the first level group to group related events.

`labels.job`: (Optional) If set, will be mapped to the `SiteName` field in the ObjectServer `alerts.status` table which will be used as the sub-group to group related events.

Sample alert-rules CRD

Note: This file is also available in the included CloudPak under pak_extensions/prometheus-rules.

```
# File: netcool-rules.yaml
# Please modify these rules to monitor specific workloads,
# containers, services or nodes in your cluster
apiVersion: monitoringcontroller.cloud.ibm.com/v1
kind: AlertRule
metadata:
  name: netcool-rules
spec:
  enabled: true
  data: |-
    groups:
    - name: alertrules.rules
      rules:
      ## Sample workload monitoring rules
      - alert: jenkins_down
        expr: absent(container_memory_usage_bytes{pod_name=~".*jenkins.*"})
        for: 30s
        labels:
          severity: critical
        annotations:
          description: Jenkins container is down for more than 30 seconds.
          summary: Jenkins down
          type: Container
      - alert: jenkins_high_cpu
        expr: sum(rate(container_cpu_usage_seconds_total{pod_name=~".*jenkins.*"}[1m]))
          / count(node_cpu_seconds_total{mode="system"}) * 100 > 70
        for: 30s
        labels:
          severity: warning
        annotations:
          description: Jenkins CPU usage is {{ humanize $value}}%.
          summary: Jenkins high CPU usage
          type: Container
      - alert: jenkins_high_memory
        expr: sum(container_memory_usage_bytes{pod_name=~".*jenkins.*"}) > 1.2e+09
        for: 30s
        labels:
          severity: warning
        annotations:
          description: Jenkins memory consumption is at {{ humanize $value}}.
          summary: Jenkins high memory usage
          type: Container
      ## End - Sample workload monitoring rules.
      ## Sample container monitoring rules
      - alert: container_restarts
        expr: delta(kube_pod_container_status_restarts_total[1h]) >= 1
        for: 10s
        labels:
          severity: warning
        annotations:
          description: The container {{ $labels.container }} in pod {{ $labels.pod }}
            has restarted at least {{ humanize $value}} times in the last hour on instance
            {{ $labels.instance }}.
          summary: Containers are restarting
          type: Container
      ## End - Sample container monitoring rules.
      ## Sample node monitoring rules
      - alert: high_cpu_load
```

```
        expr: node_load1 > 1.5
        for: 30s
        labels:
          severity: critical
        annotations:
          description: Docker host is under high load, the avg load 1m is at {{ $value}}.
            Reported by instance {{ $labels.instance }} of job {{ $labels.job }}.
          summary: Server under high load
          type: Server
      - alert: high_memory_load
        expr: (sum(node_memory_MemTotal_bytes) - sum(node_memory_MemFree_bytes +
node_memory_Buffers_bytes
          + node_memory_Cached_bytes)) / sum(node_memory_MemTotal_bytes) * 100 > 85
        for: 30s
        labels:
          severity: warning
        annotations:
          description: Docker host memory usage is {{ humanize $value}}%. Reported by
            instance {{ $labels.instance }} of job {{ $labels.job }}.
          summary: Server memory is almost full
          type: Server
      - alert: high_storage_load
        expr: (node_filesystem_size_bytes{fstype="aufs"} -
node_filesystem_free_bytes{fstype="aufs"})
          / node_filesystem_size_bytes{fstype="aufs"} * 100 > 85
        for: 30s
        labels:
          severity: warning
        annotations:
          description: Docker host storage usage is {{ humanize $value}}%. Reported by
            instance {{ $labels.instance }} of job {{ $labels.job }}.
          summary: Server storage is almost full
          type: Server
      - alert: monitor_service_down
        expr: up == 0
        for: 30s
        labels:
          severity: critical
        annotations:
          description: Service {{ $labels.instance }} is down.
          summary: Monitor service non-operational
          type: Service
      ## End - Sample node monitoring rules.
```

6. Create a new `AlertRule` in the `kube-system` namespace using the following command:

   ```
   kubectl apply -f netcool-rules.yaml --namespace kube-system
   ```

   Note : For ICP 3.1.2 or lower, the alert rules should be added in the `alert-rules` Configmap in the `kube-system` namespace instead of creating a `AlertRule` resource.

7. Prometheus usually takes a couple of minutes to reload the updated config maps and apply the new configuration. Verify that Prometheus events appear on the OMNIbus Event List.

Integrating Logstash with Netcool Operations Insight

To modify the default Logstash configuration, use the following steps:

1. After a successful deployment, get the probe's `Endpoint Host` and `Port` from the **Workloads > Deployments** page.

   - If **`logstashPobe.service.type`** is set to `ClusterIP`, the full webhook URL will have the following format:

     ```
     http://<service name>.<namespace>:<externalPort>/probe/webhook/logstash
     ```

     To obtain the service name and port using the command line, use the following commands substituting <namespace> with the namespace where the release is deployed and <release_name> with the Helm release name.

     ```
     # Get the Service name

     export SVC_NAME=$(kubectl get services --namespace <namespace> -l
     "app.kubernetes.io/instance=<release_name>,app.kubernetes.io/
     component=logstashprobe" -o jsonpath="{.items[0].metadata.name}")
     ```

```
# Get the Service port number

export SVC_PORT=$(kubectl get services --namespace <namespace> -l
"app.kubernetes.io/instance=<release_name>,app.kubernetes.io/
component=logstashprobe" -o jsonpath="{.items[0].spec.ports[0].port}")
```

- If **logstashPobe.service.type** is set to Nodeport, the full webhook URL will have the following format:

  ```
  http://<External IP>:<Node Port>/probe/webhook/logstash
  ```

  To obtain the NodePort number using the command line, use the following commands substituting <namespace> with the namespace where the release is deployed and <release_name> with the Helm release name.

  ```
  # Get the NodePort number from the Service resource

  export NODE_PORT_LOGSTASH=$(kubectl get services --namespace <namespace> -l
  "app.kubernetes.io/instance=<release_name>,app.kubernetes.io/
  component=logstashprobe" -o jsonpath="{.items[0].spec.ports[0].nodePort}")

  # On ICP 3.1.1 or later, you can obtain the External IP from the IBM Cloud
  Cluster Info Configmap using the command below.

  export NODE_IP_LOGSTASH=$(kubectl get configmap --namespace kube-public
  ibmcloud-cluster-info -o jsonpath="{.data.proxy_address}")

  echo http://$NODE_IP_LOGSTASH:$NODE_PORT_LOGSTASH/probe/webhook/logstash
  ```

2. Determine the Logstash Pipeline config map in the same namespace. In this procedure, the ConfigMap in the kube-system namespace is logging-elk-logstash-config. If a separate Logstash is deployed, determine the pipeline ConfigMap and add a new http output.

   Note : In ICP 3.1.2 or below, the Logstash Pipeline ConfigMap name is logging-elk-logstash-config.

3. Edit the Logstash pipeline ConfigMap to add a new http output. To do this using the command line, configure the kubectl client and follow the steps below.

4. Load the ConfigMap into a file using the following command:

   ```
   kubectl get configmap logging-elk-logstash-config --namespace=kube-system -o
   yaml > logging-elk-logstash-config.yaml
   ```

5. Edit the logging-elk-logstash-config.yaml file. Modify the output object to add a new http output object as shown below. Use the full webhook URL as shown in Step 2 in the http.url parameter.

   ```
   output {
         elasticsearch {
           index => "logstash-%{+YYYY.MM.dd}"
           hosts => "elasticsearch:9200"
         }
          http {
            url => "http://<ip_address>:<port>/probe/webhook/logstash"
            format => "json"
            http_method => "post"
            pool_max_per_route => "5"
          }
       }
   ```

   Note : (Optional) **pool_max_per_route** is set to limit concurrent connections to the probe to 5 so that Logstash does not flood the probe which may cause event loss.

6. Save the changes in the file and replace the ConfigMap.

   ```
   kubectl replace --namespace kube-system logging-elk-logstash-pipeline-config -f logging-elk-
   logstash-pipeline-config.yaml
   configmap "logging-elk-logstash-pipeline-config" replaced
   ```

7. Logstash takes a minute or so to reload the new configration. Check the logs to make sure there are no errors sending HTTP POST notifications to the probe.

Integrating IBM Cloud Event Management (CEM) with Netcool Operations Insight

The integration between IBM CEM and NOI requires configuration settings on the Netcool/OMNIbus ObjectServer and CEM.

Configuring the ObjectServer

This section only applies when preparing an on-premise ObjectServer with the required fields. For NOI on ICP, you can skip this section because the required fields are already pre-installed in the ObjectServer.

To integrate with IBM CEM, there are several additional ObjectServer fields required by the probe to map CEM event attributes into the `alerts.status` table. The additional fields can be added using `$OMNIHOME/bin/nco_sql`, and must be done prior to installing the chart using the following steps:

1. Copy and save the following Structured Query Language (SQL) commands into a file named `cem.sql`:

```
-- Filename: cem.sql
-- This SQL file adds the fields required by IBM Cloud Event Management
ALTER TABLE alerts.status ADD COLUMN CEMSubscriptionID VARCHAR(64);
go

ALTER TABLE alerts.status ADD COLUMN CEMIncidentUUID VARCHAR(64);
go

ALTER TABLE alerts.status ADD COLUMN NodeType VARCHAR(64);
go

ALTER TABLE alerts.status ADD COLUMN CEMEventId  VARCHAR(64);
go

ALTER TABLE alerts.status ADD COLUMN CEMErrorCode INTEGER;
go

ALTER TABLE alerts.status ADD COLUMN CEMDeduplicationKey  VARCHAR(64);
go
```

2. Load the SQL commands contained in the `cem.sql` file using the following command:

```
$OMNIHOME/bin/nco_sql -server server_name \
-user user_name \
-password 'password' < cem.sql
```

Configuring IBM CEM

To integrate and receive events from IBM Cloud Event Management, the `ibm-cem` chart must be installed in the same cluster.

For detailed steps on installing and configuring IBM Cloud Event Management in IBM Cloud Private (ICP), see Installing in IBM Cloud Private.

Use the following steps to create an outgoing integration, register the probe's webhook endpoint, and create an event forwarding policy in IBM CEM.

1. Configure and install the `ibm-netcool-probe` chart in ICP. Ensure that the `cemProbe.enabled` parameter is set to `true` to enable a probe endpoint for CEM. For more details, refer to the chart's configuration section.
2. Follow the instructions in the Notes section after installing the chart to obtain the probe's webhook endpoint URL. From the UI, go to **Menu -> Monitor Health -> Helm Releases -> (Probe release name)**
3. As a CEM Administrator, go to **Menu -> Workload -> Brokered Services -> <CEM Instance Name>** and click **Launch** to launch and log in to the CEM UI.
4. Go to the Administration page and click **Integrations**.

5. Click the **Configuring an integration** button. Select an **Outgoing** integration for Netcool/OMNIbus. Click **Configure** and use the following steps to configure the outgoing integration.
   a. Provide a name for this outgoing integration. You can use the probe release name for easy reference.
   b. You can skip **Step 2. Download and decompress a package of configuration files** because this probe chart has the required configuration and rules files preconfigured.
   c. Enter the probe webhook endpoint obtained from the probe release in the previous step.
   d. Skip **Step 4. Enter your credentials** because this step is not applicable when integrating with a probe in ICP.
   e. Turn on the integration.
   f. Click **Save** and verify that the outgoing integration is created successfully.
6. To create an event forwarding policy, go to the Administration page and click **Policies**, then click **Create event policy** to create a new event policy and use the following steps to configure the event policy. You may add a forwarding rule in an existing policy if necessary.
   a. Give a name and description for this policy.
   b. Select **All Events** to forward all CEM events to Netcool/OMNIbus. Optionally, you may configure the policy to only forward selected events.
   c. Check the **Forwarding events** option, click the **Add integrations** button and then select the outgoing integration created in the previous step.
   d. Enable this event policy.
   e. Click **Save** and verify that the event policy is successfully installed. It should be listed in the **Policies** page.
7. It may take a moment before CEM starts to forward events to Netcool/OMNIbus. Verify that CEM events appear in the Netcool/OMNIbus Event List.

Limitations

This solution has the following limitations:

- Only the AMD64 / x86_64 architecture is supported for IBM Tivoli Netcool/OMNIbus Message Bus Probe.
- Logstash integration is verified on IBM Cloud Private 3.2.0 and IBM Cloud Private 3.2.0 on Red Hat OpenShift 3.11.
- CEM and Prometheus integrations are verified on Red Hat OpenShift Container Platform 4.2 with IBM Cloud Platform Common Services 3.2.3 and Red Hat OpenShift Container Platform with IBM Cloud Platform Common Services 3.2.4.
- There are several known limitations when enabling a secure connection between probe clients and the server:
  - The required files in the secret must be created using the `nc_gskcmd` utility.
  - If your ObjectServer is configured with FIPS 140-2, the password for the key database file (`omni.kdb`) must meet the requirements stated in the following IBM Knowledge Center page: Creating a key database using nc_gskcmd.
  - When encrypting a string value using the encryption config key file (`encryption.keyfile`), you must use AES_FIPS as the cipher algorithm. AES is not supported.
  - When connecting to an ObjectServer in the same cluster, you may connect the probe to the secure connection proxy which is deployed with the IBM Netcool Operations Insight chart to encrypt the communication using TLS but the TLS termination is done at the proxy. It is recommended to enable Mutual Transport Layer Security (mTLS) in Service Mesh on OCP to secure cluster data network communications.
- Secure connection with external event sources (outside of the cluster) through Ingress is not supported.

Troubleshooting

The following table describes how to troubleshoot issues when deploying the chart and how to resolve them.

Table 2. Problems

Problem	Cause	Resolution
Probe logs show an error when loading or reading rules files. Failed during field verification check. Fields `SiteName` and `ScopeID` not found	The OMNIbus ObjectServer event grouping automation is not installed, hence the required fields are missing.	Install the event grouping automation in your ObjectServer and redeploy the chart.
The following error occurred when deploying the chart with PodDisruptionBudget enabled: `poddisruptionbudgets.policy is forbidden`	The user deploying the chart does not have the correct role to deploy the chart with **`PodDisruptionBudget`** enabled.	Administrator or Cluster Administrator role is required to deploy the chart with **`PodDisruptionBudget`** enabled.
The Logstash probe no longer receives any kubelet events from Logstash.	Since Kubernetes 1.8, kubelet writes to `journald` for systems with `systemd` instead of logging to file in a directory monitored by Logstash. So the kubelet logs are not collected by Logstash and not forwarded to the probe.	This is a known limitation and there is no resolution for this issue because it is a change in architecture.
The probe deployment failed to mount the ConfigMaps or some object name appears to be somewhat random.	If a long release name is used, the chart will generate a random suffix for objects that exceed the character limit. This may cause mapping issues between the Kubernetes objects.	Use a shorter release name, below 20 characters.
Warning messages such as `This chart requires a namespace with a ibm-restricted-psp pod security policy` are always displayed when installing the chart using the Catalog on ICP on OCP platforms.	Support for SCCs is not currently implemented for the Catalog.	These warning messages can be ignored. The chart can still to be installed using the Catalog and will apply SCCs instead of PSPs on ICP on OCP platforms. The Catalog will add support for SCCs in a future release of ICP.

Environment locale setting

You can set the locale settings for the probe process by setting the `probe.locale` parameter which will set the `LC_ALL` environment variable in the container to any of the locale settings available in the image as listed below:

- C
- POSIX
- en_US
- en_US.iso88591

- en_US.iso885915
- en_US.utf8

The default locale setting is `en_US.utf8`.

Known issues

This topic describes known issues with the helm chart.

Receiving Logstash events

The probe does not receive Logstash events due to a known issue with ICP on the following platforms:

- ICP 3.1.2 on OCP 3.10
- ICP 3.2 on OCP 3.11

For more details see:

https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/getting_started/known_issues.html for known issues on IBM Cloud Private 3.1.2.

and

https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.2.0/getting_started/known_issues.html for known issues on IBM Cloud Private 3.2.0.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA

3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, AIX, Tivoli, zSeries, and Netcool are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Adobe, Acrobat, Portable Document Format (PDF), PostScript, and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

SC27-9579-03