IBM Tivoli Netcool/OMNIbus Probe for Kubernetes Helm Chart 3.0.0

Reference Guide
February 28, 2019

SC27-8791-02

Note

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 15.

Edition notice

This edition (SC27-8791-02) applies to version 3.0.0 of IBM Tivoli Netcool/OMNIbus Probe for Kubernetes Helm Chart and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-8791-01.

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 Kubernetes 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
SC27-8791-00	May 4, 2018	First IBM publication.
SC27-8791-01	August 9, 2018	Guide updated for version 2.0 of the helm chart. "Obtaining the PPA package" on page 1 updated. "Prerequisites" on page 2 updated. Descriptions for the following parameters added to "Configurable parameters" on page 5: • **`logstashProbe.enabled`** • **`prometheusProbe.enabled`** "Limitations" on page 12 updated.
SC27-8791-02	February 28, 2019	Guide updated for version 3.0.0 of the helm chart. Helm chart now supports ICP 3.1.x. The following topics were updated: • "Obtaining the PPA package" on page 1 • "Prerequisites" on page 2 • "Resources required" on page 2 • "Configurable parameters" on page 5 • "Troubleshooting" on page 13 The following topic was added: • "PodSecurityPolicy requirements" on page 2

Chapter 1. Probe for Kubernetes Helm Chart

The Probe for Kubernetes Helm Chart allows you to deploy a cluster of Probes for Message Bus onto Kubernetes. These probes process events and alerts from Logstash HTTP output and Prometheus Alertmanager to a Netcool Operations Insight (NOI) operational dashboard.

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: `CC0F8EN`

Chart details

The chart deploys two probes onto Kubernetes which start two webhook endpoints to receive notifications in the form of HTTP `POST` requests from Logstash and Prometheus Alert Manager. 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 Tivoli Netcool/OMNIbus ObjectServer to be created and running prior to installing the probe. To create and run the IBM Tivoli Netcool/OMNIbus ObjectServer, see the following topic on the IBM Knowledge Center: Creating and running ObjectServers.
- Scope-based Event Grouping automation to be installed and enabled, see the following installation instructions on the IBM Knowledge Center: Installing scope-based event grouping.
- Kubernetes 1.11.1.
- Tiller 2.9.1
- Logstash 5.5.1.
- Prometheus 2.3.1.
- Prometheus Alert Manager 0.15.0.

Note : Operator 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 sensitive information from a secret such as TLS certificate.

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. For details see “PodSecurityPolicy requirements” on page 2.

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/SSSHTQ/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

- From the user interface, you can copy and paste the following snippets to enable the custom PodSecurityPolicy:
  - Custom PodSecurityPolicy definition:

```
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  annotations:
    kubernetes.io/description: "This policy is based on the most restrictive policy,
    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
  name: ibm-netcool-probe-psp
spec:
  allowPrivilegeEscalation: false
  forbiddenSysctls:
  - '*'
  fsGroup:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  hostNetwork: false
  hostPID: false
  hostIPC: false
  requiredDropCapabilities:
  - ALL
  runAsUser:
    rule: MustRunAsNonRoot
  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:
  - extensions
  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`

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” on page 5.

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.

`post-delete/clusterAdministration/deleteSecurityClusterPrereqs.sh`

As a Cluster Administrator, run the namespace administration cleanup script included under `pak_extensions` to clean up namespace scoped resources when appropriate.

`post-delete/namespaceAdministration/deleteSecurityNamespacePrereqs.sh`

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).

The following topics describe who to configure the integration.

Configurable parameters

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`
image.tag	The image tag. The default value is `9.0.9`
image.testRepository	Utility image repository. Update this repository name to pull from a private image repository. The default is `busybox`.
image.testImageTag	Utility image tag. The default is `1.28.4`.
image.pullPolicy	The image pull policy. The default value is `IfNotPresent`
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`
netcool.primaryServer	The primary Netcool/OMNIbus server to connect to. The default value is `nil`
netcool.primaryHost	The host of the primary Netcool/OMNIbus server. The default value is `nil`
netcool.primaryPort	The port of the primary Netcool/OMNIbus server. The default value is `nil`

Parameter name	Description
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`
netcool.backupHost	The host of the backup Netcool/OMNIbus server. The default value is `nil`
netcool.backupPort	The port of the backup Netcool/OMNIbus server. The default value is `nil`
logstashProbe.enabled	Set this parameter to `true` to enable the probe for Logstash. The default value is `true`
logstashProbe.replicaCount	The number of deployment replicas of the Logstash probe. This will be ignored if `logstashProbe.autoscaling.enabled=true` and will use the **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`

Parameter name	Description
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`
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`

Parameter name	Description
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
probe.messageLevel	The probe log message level. The default value is `warn`
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 Fixed to amd64.

Integrating Prometheus Alert Manager with Netcool Operations Insight

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

1. Deploy the `ibm-netcool-probe` chart.
2. After a successful deployment, get the Prometheus 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/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 **`logstashPobe.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 ICP 3.1.1, 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

# On ICP 3.1.0, get the External IP from the Nodes resource. This command
requires Cluster Administrator role. export NODE_IP_PROMETHEUS=$(kubectl
get nodes -l proxy=true -o
jsonpath="{.items[0].status.addresses[0].address}")
```

3. Determine the Prometheus Alert Manager and Alert Rules config maps in the same namespace. In this procedure, the config maps in the `kube-system` namespace are `monitoring-prometheus-alertmanager` and `alert-rules` respectively.
4. Edit the Prometheus Alert Manager pipeline 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
   ```

5. 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.
6. 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
   ```

7. Load the `alert-rules` ConfigMap into a file, update the data section to add your alerting rules and save the file. Sample rules for Prometheus 2.0 or newer are shown below.

```
$ kubectl get configmap monitoring-prometheus-alert-rules --namespace=kube-
system -o yaml > alertrules.yaml
```

```
$ cat alertrules.yaml
apiVersion: v1
data:
  alert.rules: |-
    groups:
    - name: alertrules.rules
      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
      - 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
      - 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
      - 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
      - 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
      - alert: monitor_service_down
```

```
        expr: up == 0
        for: 30s
        labels:
          severity: critical
        annotations:
          description: Service {{ $labels.instance }} is down.
          summary: Monitor service non-operational

kind: ConfigMap
metadata:
  creationTimestamp: 2018-04-18T02:38:14Z
  labels:
    app: monitoring-prometheus
    chart: ibm-icpmonitoring-1.3.0
    component: prometheus
    heritage: Tiller
    release: monitoring
  name: monitoring-prometheus-alertrules
  namespace: kube-system
  resourceVersion: "1856491"
  selfLink: /api/v1/namespaces/kube-system/configmaps/monitoring-prometheus-alertrules
  uid: 8aee6593-42b1-11e8-bd3d-0050569b6c73
```

8. Replace the ConfigMap with the updated configuration using the following command:

   ```
   kubectl replace confimap monitoring-prometheus-alertrules --namespace=kube-system -f alertrules.yaml
   ```

   ```
   configmap "monitoring-prometheus-alertrules" replaced
   ```

9. 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. Deploy the `ibm-netcool-probe` chart.
2. After a successful deployment, get the Logstash 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, 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}")

# On ICP 3.1.0, get the External IP from the Nodes resource. This command
requires Cluster Administrator role. export NODE_IP_LOGSTASH=$(kubectl get
nodes -l proxy=true -o jsonpath="{.items[0].status.addresses[0].address}")
```

3. 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`.
4. 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.
5. 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
   ```

6. 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.

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

   ```
   kubectl replace --namespace kube-system logging-elk-logstash-config -f logging-elk-logstash-
   config.yaml

   configmap "logging-elk-logstash-config" replaced
   ```

8. 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.

Limitations

This solution has the following limitations:

- Only the AMD64 / x86_64 architecture is supported for IBM Tivoli Netcool/OMNIbus Message Bus Probe.
- It is validated to run on IBM Cloud Private 3.1.0 and 3.1.1.

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.

Known issues

There are currently no known issues with the helm chart.

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.

SC27-8791-02