Kubernetes Administration Guide

FortiOS 7.2

FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO GUIDE
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/training-certification

NSE INSTITUTE
https://training.fortinet.com

FORTIGUARD CENTER
https://www.fortiguard.com

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdoc@fortinet.com

FORTINET®

March 31, 2022
FortiOS 7.2 Kubernetes Administration Guide
01-720-793045-20220331

TABLE OF CONTENTS

About FortiGate-VM and Kubernetes

FortiOS supports automatically updating dynamic addresses for Kubernetes (K8s) using a K8s SDN connector, enabling FortiOS to manage K8s pods as global address objects, as with other connectors.

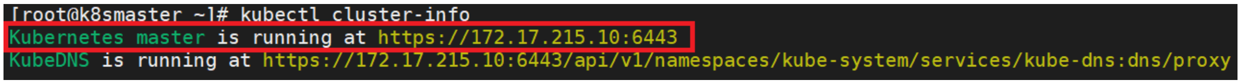
In addition, Fortinet has partnered with Tigera for further integration between Calico and FortiGate. Calico and Calico Enterprise provide the networking and security framework to secure K8s networks. The largest public cloud providers have selected Calico to provide network security for their hosted K8s services. Through its Firewall Manager integration, it can offload zone-based security to the FortiGate firewall. This is accomplished by providing dynamic address updates directly to the FortiGate via REST API.

Obtaining the IP address, port, and secret token in Kubernetes

Configuring a Kubernetes (K8s) private cloud SDN connector in FortiOS requires the IP address and port that the K8s deployment is running on, as well as an authentication token.

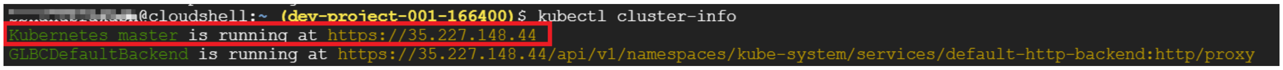
To obtain the IP address, port, and secret token in K8s:

1. When configuring the K8s SDN connector in FortiOS, you must provide the IP address and port that the K8s deployment is running on. Run `kubectl cluster-info` to obtain the IP address and port. Note down the IP address and port. The following shows the IP address and port for a local cluster:

   ```
   [root@k8smaster ~]# kubectl cluster-info
   Kubernetes master is running at https://172.17.215.10:6443
   KubeDNS is running at https://172.17.215.10:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
   ```

   The following shows the IP address and port for customer-managed K8s on Google Cloud Platform:

   ```
   [redacted]@cloudshell:~ (dev-project-001-166400)$ kubectl cluster-info
   Kubernetes master is running at https://35.227.148.44
   GLBCDefaultBackend is running at https://35.227.148.44/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
   ```

2. Generate the authentication token:
   a. Create a service account to store the authentication token:
      i. Run the `kubectl create serviceaccount <Service_account_name>` command. For example, if the service account name is fortigateconnector, the command is `kubectl create serviceaccount fortigateconnector`.
      ii. Run the `kubectl get serviceaccounts` command to verify that you created the service account. The account should show up in the service account list.
   b. Create a cluster role. K8s 1.6 and later versions allow you to configure role-based access control (RBAC). RBAC is an authorization mechanism to manage resource permissions on K8s. You must create a cluster role to grant the FortiGate permission to perform operations and retrieve objects:
      i. Create the yaml file by running the `vi <filename>.yaml` command. For example, if the yaml file name is fgtclusterrole, the command is `vi fgtclusterrole.yaml`. Paste the following:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
# "namespace" omitted since ClusterRoles are not namespaced
```

```
 name: fgt-connector

rules:
- apiGroups: [""]

  resources: ["pods", "namespaces", "nodes" , "services"]
  verbs: ["get", "watch", "list"]
```

The resources list specifies the objects that FortiOS can retrieve. The verbs list specifies the operations that FortiOS can perform.

ii. Run the `Kubectl apply -f <filename>.yaml` command to apply the yaml file to create the cluster role. In this example, the command is `Kubectl apply -f fgtclusterrole.yaml`.

iii. Run the `kubectl create clusterrolebinding fgt-connector --clusterrole=<cluster_rolename> --serviceaccount=default:<service_account_name>` to attach the cluster role to the service account. In this example, the command is `kubectl create clusterrolebinding fgt-connector --clusterrole=fgt-connector --serviceaccount=default:fortigateconnector`.

c. Run the `kubectl get secrets -o jsonpath="{.items[?(@.metadata.annotations['kubernetes\.io/service-account\.name']=='fortigateconnector')].data.token}"| base64 --decode` command to obtain the secret token. As the token is Base64 encoded, the command includes `base64 --decode` to extract the decoded keystring. Note down the token.

Automatically updating dynamic addresses using Calico FortiGate integration

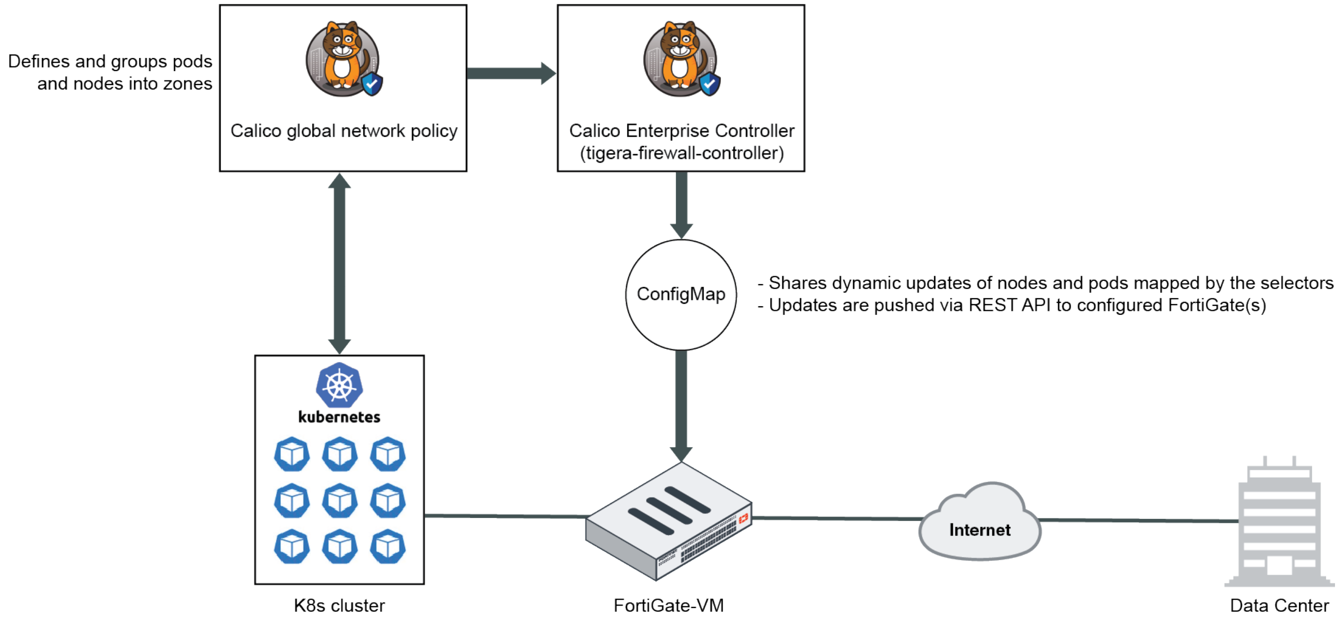
When deploying a Kubernetes (K8s) cluster, you can install a third-party network policy provider. Calico is a popular provider that provides the necessary framework to protect and secure the network. The largest public cloud providers have selected Calico to provide network security for their hosted K8s services (Amazon EKS, Azure AKS, Google GKE, and IBM IKS) running across tens of thousands of clusters.

Through its Firewall Manager integration, Calico can effectively separate network controls and security controls. Operationally, this allows a company to assign security tasks to the Security Operations team using a familiar firewall such as the FortiGate and management tool such as FortiManager.

Nearly every application has dependencies external to K8s that require some level of access control, such as access requirements for database, third-party APIs, and cloud services.

Calico implements zone-based security in order to secure K8s workloads. For example, Internet-facing workloads run in the demilitarized zone, while other workloads for backend business logic may run in the trusted zone. These workloads are dynamic in nature and can be brought up/down and moved across nodes and clusters frequently. Therefore, a Firewall Manager must be informed of each dynamic address change to properly secure the workload.

See Extend FortiGate Firewalls to Kubernetes with Calico Enterprise in Tigera's documentation for the general workflow. Following is a high-level overview of the workflow:


Defines and groups pods and nodes into zones
Calico global network policy
Calico Enterprise Controller (tigera-firewall-controller)
ConfigMap
- Shares dynamic updates of nodes and pods mapped by the selectors
- Updates are pushed via REST API to configured FortiGate(s)
kubernetes
K8s cluster
FortiGate-VM
Internet
Data Center

The Calico Enterprise Controller, also called tigera-firewall-controller, shares K8s node and pod addresses with FortiGate. The controller uses a ConfigMap to define the selectors for mapping the workloads to firewall address groups. The ConfigMap also defines the desired FortiGate(s)/FortiManager(s) to communicate with. The controller then pushes dynamic updates to the FortiGate(s) via REST API. Subsequently, traffic from the K8s cluster passes through the FortiGate, and you can administer zone-based security using firewall policies.

To configure automatically updating dynamic addresses using Calico FortiGate integration:

1. Configure Calico assets as Extend FortiGate Firewalls to Kubernetes with Calico Enterprise describes.
2. Configure a REST API administrator in FortiOS:
   a. Go to *System* > *Administrators*, then select *Create New* > *REST API Admin*.
   b. In the *Username* field, enter a username, such as *calico_enterprise_api_user*.
   c. If desired, enter comments.
   d. Creating a new administrator profile with minimal privileges is recommended. Create a new profile:
      i. From the *Administrator Profile* dropdown list, select *Create*.
      ii. In the *Name* field, enter the desired name, such as *tigera_api_user_profile*.
      iii. Under *Access Permissions*, configure the following:
         i. For *Firewall*, select *Custom*.
         ii. For *Address*, select *Read/Write*. The REST API can send read and write requests (HTTP GET/POST/PUT/DELETE) to the resource.
         iii. For all others, leave as *None*.
      iv. Click *OK*.
   e. FortiOS displays an API key. Copy and store the key securely, as it is only shown once.

Once configuration is complete on the FortiGate and Calico, you see address objects being created on the FortiGate. When changes occur on your workloads, the address objects change as well. The address objects are marked with a “Managed by Tigera Calico Enterprise” comment.

With these new dynamic address groups, you can define firewall policies to deploy zone-based security for your K8s network.

Collecting only node IP addresses with Kubernetes SDN connectors

By default, Kubernetes SDN connectors return both pod and node IP addresses. Peer Kubernetes SDN connectors can be configured to resolve dynamic firewall IP addresses to only node IP addresses. Results can also be filtered by specific IP addresses.

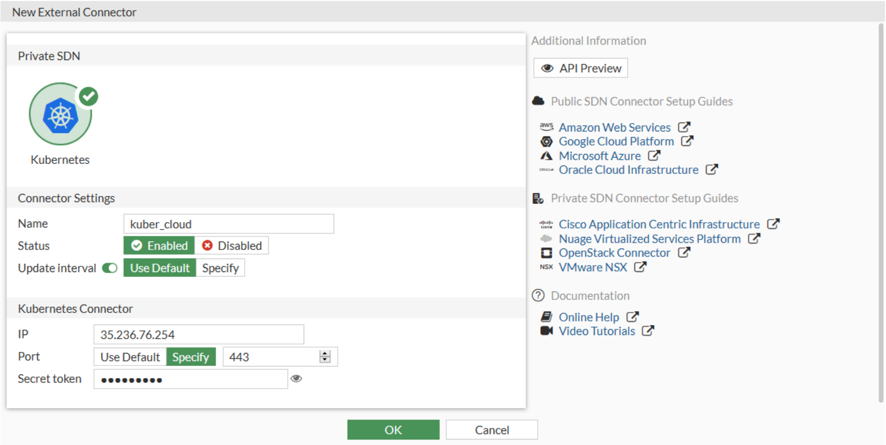
Example

In this example, a Kubernetes SDN connector and two dynamic firewall addresses are created. One of the addresses is configured to resolve only node IP addresses, while the other resolves both the pod and node IP addresses.

GUI configuration

To configure a Kubernetes SDN connector in the GUI:

1. Go to *Security Fabric > External Connectors* and click *Create New*.
2. Select Kubernetes, then configure the connector settings:



New External Connector

Private SDN

Kubernetes

Connector Settings

Name: kuber_cloud

Status: ● Enabled ○ Disabled

Update interval: Use Default | Specify

Kubernetes Connector

IP: 35.236.76.254

Port: Use Default | Specify: 443

Secret token: •••••••••

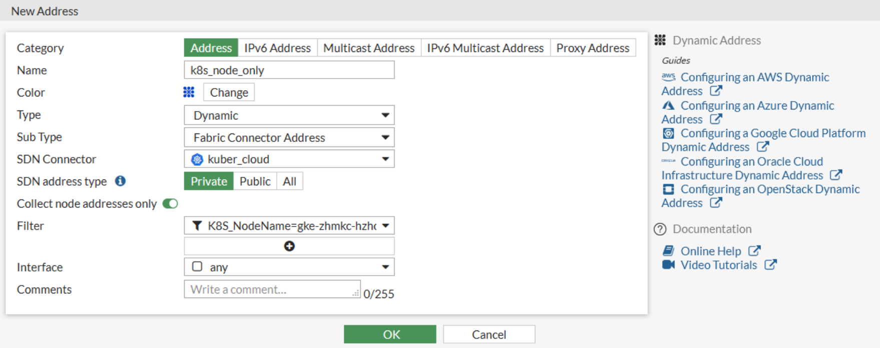
Additional Information
API Preview
Public SDN Connector Setup Guides
Amazon Web Services
Google Cloud Platform
Microsoft Azure
Oracle Cloud Infrastructure
Private SDN Connector Setup Guides
Cisco Application Centric Infrastructure
Nuage Virtualized Services Platform
OpenStack Connector
VMware NSX
Documentation
Online Help
Video Tutorials
OK
Cancel

Name	kuber_cloud
IP	35.236.76.254
Port	Specify - 443
Secret token	*********

3. Click *OK*.

To create the two dynamic firewall addresses in the GUI:

1. Go to *Policy & Objects* > *Addresses* and click *Create New* > *Address*.



New Address

Category	Address | IPv6 Address | Multicast Address | IPv6 Multicast Address | Proxy Address
Name	k8s_node_only
Color	Change
Type	Dynamic
Sub Type	Fabric Connector Address
SDN Connector	kuber_cloud
SDN address type	Private | Public | All
Collect node addresses only	●
Filter	K8S_NodeName=gke-zhmkc-hzh
Interface	□ any
Comments	Write a comment... 0/255

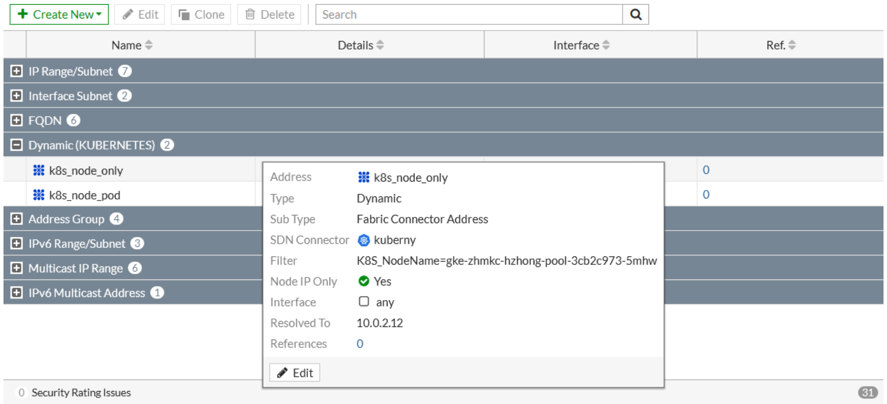
OK
Cancel
Dynamic Address
Guides
Configuring an AWS Dynamic Address
Configuring an Azure Dynamic Address
Configuring a Google Cloud Platform Dynamic Address
Configuring an Oracle Cloud Infrastructure Dynamic Address
Configuring an OpenStack Dynamic Address
Documentation
Online Help
Video Tutorials

Name	k8s_node_only
Type	Dynamic
Sub Type	Fabric Connector Address
SDN Connector	kuber_cloud
SDN address type	Private
Collect node addresses only	Enabled
Filter	K8S_NodeName=gke-zhmkc-hzhong-pool-3cb2c973-5mhw

2. Click *OK*.
3. Click *Create New* > *Address* again to create the second address.
4. Configure the same settings as the first address, except set *Name* to *k8s_node_pod* and disable *Collect node addresses only*.
5. Click *OK*.

To check the resolved IP addresses of the two dynamic addresses in the GUI:

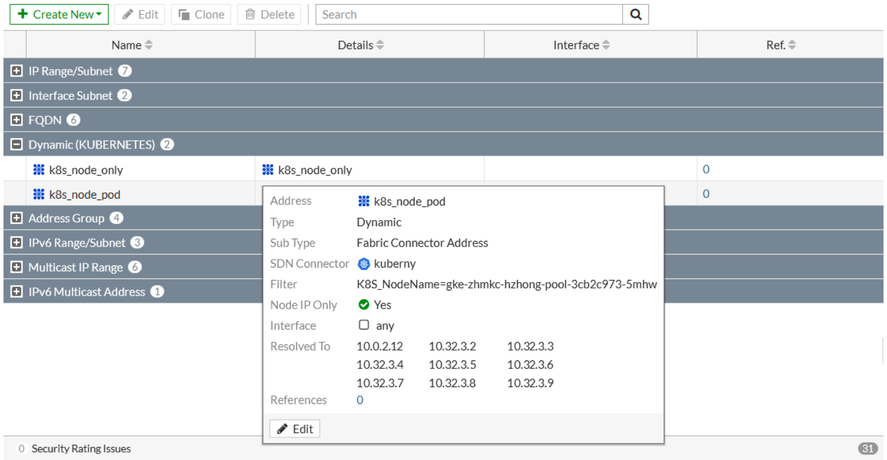
1. Go to *Policy & Objects* > *Addresses*.
2. In the address list, hover the cursor over the *k8s_node_only* address. Only the node IP address is resolved.



Name	Details	Interface	Ref.
IP Range/Subnet 7			
Interface Subnet 2			
FQDN 6			
Dynamic (KUBERNETES) 2			
k8s_node_only			0
k8s_node_pod			0
Address Group 4			
IPv6 Range/Subnet 3			
Multicast IP Range 6			
IPv6 Multicast Address 1			

Address	k8s_node_only
Type	Dynamic
Sub Type	Fabric Connector Address
SDN Connector	kuberny
Filter	K8S_NodeName=gke-zhmkc-hzhong-pool-3cb2c973-5mhw
Node IP Only	✓ Yes
Interface	□ any
Resolved To	10.0.2.12
References	0

3. Hover over the k8s_node_pod address. The node and pod IP addresses are all resolved.



Name	Details	Interface	Ref.
IP Range/Subnet 7			
Interface Subnet 2			
FQDN 6			
Dynamic (KUBERNETES) 2			
k8s_node_only	k8s_node_only		0
k8s_node_pod			0
Address Group 4			
IPv6 Range/Subnet 3			
Multicast IP Range 6			
IPv6 Multicast Address 1			

Address	k8s_node_pod
Type	Dynamic
Sub Type	Fabric Connector Address
SDN Connector	kuberny
Filter	K8S_NodeName=gke-zhmkc-hzhong-pool-3cb2c973-5mhw
Node IP Only	✓ Yes
Interface	□ any

Resolved To	10.0.2.12 10.32.3.2 10.32.3.3
	10.32.3.4 10.32.3.5 10.32.3.6
	10.32.3.7 10.32.3.8 10.32.3.9

References	0

0 Security Rating Issues 31

The resolved IP addresses can be verified by accessing the Kubernetes cluster directly, see Verify the resolved IP addresses on page 10.

CLI configuration

To configure a Kubernetes SDN connector in the CLI:

```
config system sdn-connector
    edit "kuber_cloud"
        set type kubernetes
        set server "35.236.76.254"
        set server-port 443
        set secret-token *********
    next
end
```

To create the two dynamic firewall addresses in the CLI:

```
config firewall address
    edit "k8s_node_only"
        set type dynamic
        set sdn "kuber_cloud"
        set color 19
        set filter "K8S_NodeName=gke-zhmkc-hzhong-pool-3cb2c973-5mhw"
        set node-ip-only enable
    next
    edit "k8s_node_pod"
        set type dynamic
        set sdn "kuber_cloud"
        set color 19
        set filter "K8S_NodeName=gke-zhmkc-hzhong-pool-3cb2c973-5mhw"
        set node-ip-only disable
    next
end
```

To check the resolved IP addresses of the two dynamic addresses in the CLI:

```
#show firewall address
config firewall address
```

```
    ...
    edit "k8s_node_only"
        ...
        config list
            edit "10.0.2.12"
            next
        end
    next
    edit "k8s_node_pod"
        ...
        config list
            edit "10.0.2.12"
            next
            edit "10.32.3.2"
            next
            edit "10.32.3.3"
            next
            edit "10.32.3.4"
            next
            edit "10.32.3.5"
            next
            edit "10.32.3.6"
            next
            edit "10.32.3.7"
            next
            edit "10.32.3.8"
            next
            edit "10.32.3.9"
            next
        end
    next
end
```

The resolved IP addresses can be verified by accessing the Kubernetes cluster directly.

Verify the resolved IP addresses

To confirm the node IP address:

```
fosqa@pc56:~$ kubectl get nodes gke-zhmkc-hzhong-pool-3cb2c973-5mhw -o wide
NAME                                  STATUS   ROLES    AGE    VERSION          INTERNAL-IP
 EXTERNAL-IP     OS-IMAGE                             KERNEL-VERSION   CONTAINER-RUNTIME
gke-zhmkc-hzhong-pool-3cb2c973-5mhw   Ready    <none>   532d   v1.12.7-gke.10   10.0.2.12
 35.236.118.65   Container-Optimized OS from Google   4.14.106+        docker://17.3.2
```

To confirm the node and pods IP addresses:

```
fosqa@pc56:~$ kubectl get pods --all-namespaces -o wide | grep gke-zhmkc-hzhong-pool-
3cb2c973-5mhw
default       guestbook-qcg7j                                  1/1     Running   0
186d   10.32.3.9   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
default       redis-master-mstb4                               1/1     Running   0
186d   10.32.3.8   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
default       redis-slave-7tgcv                                1/1     Running   0
186d   10.32.3.5   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
```

```
kube-system   fluentd-gcp-scaler-6965bb45c9-2lpp2              1/1     Running   0
239d   10.32.3.4   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   fluentd-gcp-v3.2.0-nnlnp                         2/2     Running   0
239d   10.0.2.12   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   heapster-gke-7858846d4d-vqc4d                    3/3     Running   0
186d   10.32.3.6   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   kube-dns-5995c95f64-rqn4b                        4/4     Running   0
186d   10.32.3.7   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   kube-dns-autoscaler-8687c64fc-dq9fn              1/1     Running   0
239d   10.32.3.2   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   kube-proxy-gke-zhmkc-hzhong-pool-3cb2c973-5mhw   1/1     Running   0
532d   10.0.2.12   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   metrics-server-v0.3.1-5c6fbf777-7bchg            2/2     Running   0
239d   10.32.3.3   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
kube-system   prometheus-to-sd-xndgs                           2/2     Running   0
186d   10.0.2.12   gke-zhmkc-hzhong-pool-3cb2c973-5mhw   <none>           <none>
```

Change log

Date	Change Description
2022-03-31	Initial release.

www.fortinet.com