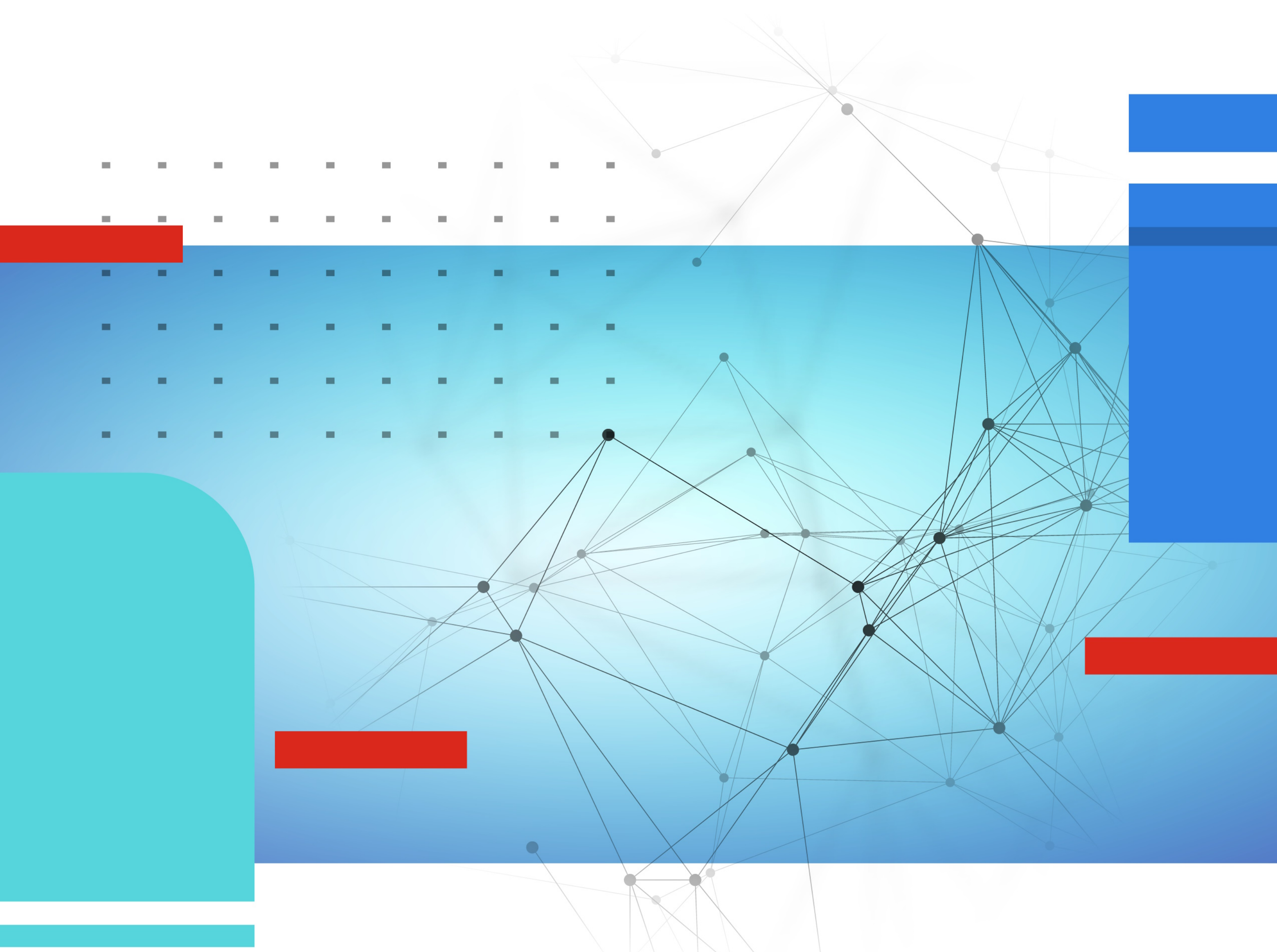
Getting Started (Kubernetes)

Container FortiOS 7.2.1



June 13, 2025
Container FortiOS 7.2.1 Getting Started (Kubernetes)
87-721-971807-20250613

TABLE OF CONTENTS

Change Log

Date	Change Description
2024-07-17	Initial release.
2025-06-13	Updated Getting the container image on page 6.

Introduction

Container FortiOS provides NGFW firewall features, including security policies, IPS inspection, application control, URL filtering, and antivirus in a container-deployed format.

It supports Linux Containers (LXC), Docker, and Kubernetes.

This guide provides information about the installation and configuration of Container FortiOS version 7.2.1, build 0255 on Kubernetes.

When Container FortiOS is deployed in a Kubernetes cluster, you can use Configmap and Secrets to manage its license and configuration.

Deploying Container FortiOS

This section provides an overview of the procedures for deployment of Container FortiOS.

As container environments vary widely, this document provides basic instructions for deployment to Kubernetes and does not provide in-depth information about configuration of Kubernetes itself.

The basic steps for deployment are as follows:

1. Ensure that all prerequisites are met.
2. Get the container image.
3. Add the image to a container registry.
4. Create role bindings.
5. Deploy the license.
6. Deploy the container.

Prerequisites

Before deploying Container FortiOS into a Kubernetes cluster, ensure that you have met the following requirements:

- A working Kubernetes cluster.
- Install Multus CNI if multiple network interfaces are needed.
  See https://github.com/k8snetworkplumbingwg/multus-cni.
- Kubernetes `kubectl` command line tool installed.
  See https://kubernetes.io/docs/reference/kubectl/.

Getting the container image

After purchasing a Container FortiOS license, submit a ticket through the Customer Service & Support site. The Technical Assistance center (TAC) team will then provide you with the appropriate image file.

For more information about submitting a ticket, see Technical Tip: How to create a ticket for Fortinet TAC.

Typically, if your container engine is based on `docker` or `containerd`, download the Docker image which will have the following naming convention:

```
FOS_<CPU Arch>_<Container Type>-v<Major Version>-build<build number>-<Company>.tar.gz
```

For example, image `FOS_X64_DOCKER-v7-build0255-FORTINET.tar.gz` was built for Docker running on an 64 bit Intel CPU device. The major version is 7 and build number is 0255.

Add the image to a container registry

After you have download the container image, add it to your container registry.

Creating role bindings

Container FortiOS needs to read `ConfigMaps` and `Secrets` and watch for changes.

Give proper permissions to the service account running Container FortiOS with `RoleBindings`.

The following example `rolebingdings.yaml` file shows the use of role bindings to allow the service account to get, watch, and list `ConfigMaps` and `Secrets`:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    namespace: default
    name: configmap-reader
rules:
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "watch", "list"]

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
    name: read-configmaps
    namespace: default
subjects:
  - kind: ServiceAccount
    name: default
    apiGroup: ""
roleRef:
    kind: ClusterRole
    name: configmap-reader
    apiGroup: ""

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    namespace: default
    name: secrets-reader
rules:
  - apiGroups: [""] # "" indicates the core API group
    resources: ["secrets"]
    verbs: ["get", "watch", "list"]

---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
    name: read-secrets
    namespace: default
subjects:
  - kind: ServiceAccount
    name: default
    apiGroup: ""
roleRef:
    kind: ClusterRole
    name: secrets-reader
    apiGroup: ""
```

In this example, the `RoleBinding` is applied for the `ServiceAccount` named `default`. Change this to an appropriate value for your system.

For more information about role binding, see https://kubernetes.io/docs/reference/access-authn-authz/rbac/.

To apply the role binding configuration, enter the following command:

```
kubectl apply -f rolebindings.yaml
```

Deploying a license

1. Use `ConfigMap` to deploy the Container FortiOS license file.
   The below example `license-configmap.yaml` file shows this configuration:

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
       name: fos-license
       labels:
           app: fos
           category: license
   data:
       license: |
           -----BEGIN FGT VM LICENSE-----
           QAAAAMWKpwAVtdp+7DxYWGkQUv5N05GWx292S+/gMhYGkK0b/prV3GRDA8umTdd3
           PhW40EGSupYYV2Lg2+bE6VGID33gGQAAbR9yMHkcF6E3aCToGZ5i90DeGdvnGjfr
           VZ+6izjyg9h9Cg/T11+1BMZOAhkyKspod6WcEfQG/jhT0cCV8NXf9dbUKpo/3Rt2
           ...............
           -----END FGT VM LICENSE-----
   ```

   The labels `app: fos` and `category: license` are required.
2. Replace the license data with the full contents of your license file.
3. Deploy the license with the following command:

   ```
   kubectl apply -f license-configmap.yaml
   ```

Deploying the container

1. Prepare a deployment YAML file.

   The below `cfos.yaml` file provides an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: fos-deployment
    labels:
        app: fos
spec:
    replicas: 1
    selector:
        matchLabels:
            app: fos
    template:
        metadata:
            labels:
                app: fos
        spec:
            containers:
              - name: fos
                image: <image_URL>
                   securityContext:
                       capabilities:
                           add: ["NET_ADMIN"]
                   ports:
                     - name: isakmp
                       containerPort: 500
                       protocol: UDP
                     - name: ipsec-nat-t
                       containerPort: 4500
                       protocol: UDP
                   volumeMounts:
                     - mountPath: /data
                       name: data-volume
               volumes:
                 - name: data-volume
                   emptyDir: {}
```

2. Update the values according to the following guidelines:
   - Labels: All resources used by Container FortiOS should be labeled `app:fos`.
   - Capabilities: Container FortiOS requires `NET_ADMIN` capability added as it uses `iptables` rules.
   - Ports: This example exposes ports `500` and `4500` for IPsec.
   - Volumes: Container FortiOS needs a volume mounted to /data/ as persistent storage for configurations and logs. In this example, a ramdisk is used to simplify the installation. When Container FortiOS boots, it imports configurations stored in the `ConfigMap` so configurations are not lost when the Container FortiOS pod is replaced.
3. Deploy Container FortiOS with the following command:

```
kubectl apply -f cfos.yaml
```

Troubleshooting

When Container FortiOS receives the `ConfigMap` object and applies the configuration, it generates logs. Use the `kubectl logs` command to view them.

For example, the following is a command to view the logs for a container with label `app=fos`.

```
kubectl logs --tail=200 -l app=fos
```

For more information, see https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#logs.

Container FortiOS also supports syslogd for sending logs to an syslogd server, such as FortiAnalyzer.

Using Container FortiOS

This section provides an overview of the initial steps for connecting to and configuring the running Container FortiOS container.

Connecting to the Container FortiOS CLI

Container FortiOS provides access to the FortiOS shell for CLI usage as well as the underlying Linux shell.

The Container FortiOS CLI is based on the FortiOS CLI, but has fewer available options.

To connect to the running Container FortiOS container:

In the host shell, enter the following command:

```
kubectl exec --stdin --tty <container_name> -- /bin/cli
```

The initial username is `admin` with an empty password. Use `config system admin` to set a password.

To enter the Linux shell, use the following command:

```
sysctl sh
```

Deploying configurations to Kubernetes

In Kubernetes, configurations can be applied to Container FortiOS using a `ConfigMap`.

The two types of configuration are as follows:

- Partial configuration: A partial configuration is applied on top of a current configuration in Container FortiOS. A configuration can be split into multiple smaller configurations and applied separately.
- Full configuration: The active configuration will be replaced with the new configuration.

Deploying a partial configuration

The following example ipsec-configmap.yml file shows a partial configuration:

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: foscfg-ipsec
    labels:
        app: fos
        category: config
```

```
data:
    type: partial
    config: |-
        config vpn certificate ca
            edit "ipsec-ca"
                set ca "-----BEGIN CERTIFICATE-----
                MIIDJDCCAgygAwIBAgIJAK6dHv+qKBjJMA0GCSqGSIb3DQEBCwUAMBExDzANBgNV
                BAMMBnRlc3RjYTAeFw0yMjAxMTMxODIxMThaFw0zMjAxMTExODIxMThaMBExDzAN
                BgNVBAMMBnRlc3RjYTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOXE
                ct+WmzZ8YT+rJEQKDGfgqiJu9kzNz+Na0smwPvFEOfcm6XYHqy/li+CdyIGCtlQX
                hDbABD7uQiVBObzO4VzPn3Ik7PMR+hBr0sULqOQ8SkgU/H/pgm5WjSO0oiiPoQon
                LWDQXs294aF0EouNp0KfI9vXkAvzv57RUGeuPfr9tvoLyIgBB1nqWbK98GfMyX1K
                sHB0mp0PCxq1S6hQK9pny3/wvsq3YxggpJAFpCAbDXI97jhk9atMaIRjGErUZNsO
                .....
                .....
                .....
                -----END CERTIFICATE-----"
                next
            end
        config vpn certificate local
            edit "ipsec-cert"
                set password "{{ipsec-certs:ipsec-cert-pass}}"
                set private-key "{{ipsec-certs:ipsec-cert-key}}"
                set certificate "-----BEGIN CERTIFICATE-----
                MIIDYDCCAkigAwIBAgIQAx0NCLIRx9Q5lWcGmS2U+DANBgkqhkiG9w0BAQsFADAR
                MQ8wDQYDVQQDDAZ0ZXN0Y2EwHhcNMjIwMTEzMTkwMDUzWhcNMjQwNDE3MTkwMDUz
                WjAYMRYwFAYDVQQDDA1pcHNlYy1jbGllbnQyMIIBIjANBgkqhkiG9w0BAQEFAAOC
                AQ8AMIIBCgKCAQEAyXXh8OiuEf5Drh+df3FJm2f/ZKNvRONEQba/77cHVRT2pjOV
                07llYQye1mG0JBedUM0SFEkmWkafyYE+KzYzse2r7NSX1bkFizW/TwrNk/VCuLMt
                +HUgClrcmrPAdbDUZYyIKWKN4FwlOyZz0YNA14NuM/gNE+fY1kaaaojxqfpneJCW
                nYcfCTuNgADnyHjzXZMLulj+4Cy1OylKSKX7cAVt9pS2SwzzGF4fGnlDKhfAtxzR
                .....
                .....
                .....
                -----END CERTIFICATE-----"
                next
            end
        config vpn ipsec phase1-interface
            edit "test-p1"
                set interface "eth0"
                set peertype any
                set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-
prfsha384 chacha20poly1305-prfsha256
                set psksecret {{ipsec-psks:psk1}}
                set auto-negotiate disable
                next
            end
        config vpn ipsec phase2-interface
            edit "test-p2"
                set phase1name "test-p1"
                set proposal aes128-sha1 aes256-sha1 aes128-sha256 aes256-sha256 aes128gcm
aes256gcm chacha20poly1305
                set dhgrp 14 15 5
                set src-subnet 10.4.96.0 255.255.240.0
                set dst-subnet 10.0.4.0 255.255.255.0
                next
            end
```

Configuration should be created with the following guidelines:

- Labels `app: fos` and `category: config` are required.
- `type: partial` indicates that this is a partial configuration.
- The `config` section holds the actual configuration data as a series of CLI commands.
- In the configuration, there are variables (for example, `{{ipsec-certs:ipsec-cert-pass}}` and `{{ipsec-certs:ipsec-cert-key}})` that are references to the keys in `Secrets`. Kubenets use `Secrets` to store sensitive data.

  In this example, we save an IPSEC pre-shared key in a `Secret` called `ipsec-certs` with key `ipsec-cert-pass`. In the configuration we can use `{{ipsec-certs:ipsec-cert-pass}}` to refer it.

  The format is `{{<Secret name>:<Key name>}}`.

  The following example command creates this secret:

  ```
  kubectl create secret generic ipsec-certs --from-literal=ipsec-cert-pass=12345678
  ```

  For more information about Kubernets `Secrets`, see https://kubernetes.io/docs/concepts/configuration/secret/.

Deploying a full configuration

Full configuration has the same format with partial configuration except `type` is `full` instead of `partial`.

The following is an example of the commands used to create a `ConfigMap` for a full configuration:

```
kubectl create configmap fos-config --from-file=config=<path to config file> --from-
literal=type=full"
kubectl label configmap fos-config app=fos
kubectl label configmap fos-config category=config
```

Ensure the configuration file contains all required dependencies. For example, if a firewall policy references a web filter profile `block-category-11`, the web filter profile and all of its dependencies must be included in the configuration.

More information

Additional Container FortiOS documentation is available in the Fortinet Documentation Library.

FortiOS documentation

Configuration and administration of Container FortiOS is very similar to FortiOS.

The following FortiOS documentation may be helpful:

- FortiOS Administration Guide
- FortiOS CLI Reference
- FortiOS REST API Reference on FNDN

www.fortinet.com