Ingress Controller Installation Guide

FortiWeb 7.4.0

Sept 8, 2023
FortiWeb 7.4.0 Ingress Controller Installation Guide
1st Edition

TABLE OF CONTENTS

Change log 4
Prerequisite knowledge 5
FortiWeb Ingress Controller Overview 6
  Supported Environments 7
  Supported Release and Version 7
  The Kubernetes API Version 8
Installation 9
  Get Repo Information 11
  Install the Helm Chart 11
  Upgrade the Helm Chart 11
  Check the Installation 12
  Uninstall the Helm Chart 13
Kubernetes Ingress 14
 Ingress class 14
 Mapping of the Ingress related resources with the FortiWeb objects 14
 Naming rule 15
 Ingress types 15
  Default backend 16
  Minimal-Ingress 17
  Name-based virtual hosting 17
  Hostname wildcards 19
  TLS 20
Configuration parameters 23
  FortiWeb Authentication Secret 23
  Annotation in Ingress 23
  Annotation in Service 25
Deployment 26
  Deploy the Pods and expose the Services 26
  Deploy the Ingress 26
  FortiView 28
  Update or delete the Ingress 29
  Add, update or delete Service and Node 29
Debug 32
FAQ 33

Change log

Sept 8, 2023	Bug fixes

Prerequisite knowledge

Kubernetes

Before you begin using FortiWeb Ingress Controller, you will need to have prerequisite knowledge of the Kubernetes cluster, and Kubernetes Ingress, Service, Pod and Node. The terms and concepts discussed in this document is sourced directly from Kubernetes official documentation. For more information, please refer to the documents listed below:

- Kubernetes Concepts: https://kubernetes.io/docs/concepts/
- Kubernetes Ingress: https://kubernetes.io/docs/concepts/services-networking/ingress/
- Kubernetes Service: https://kubernetes.io/docs/concepts/services-networking/service/

Helm Charts

As Helm Charts are used in FortiWeb Ingress Controller installation, you will also need to have understanding of how Helm Charts work. For more information, please refer to the documents listed below:

- Helm Charts values files: https://helm.sh/docs/chart_template_guide/values_files/
- Helm Charts Installation and upgrade from the Helm repository: https://helm.sh/docs/helm/helm_install/

FortiWeb Ingress Controller Overview

FortiWeb Ingress Controller fulfills the Kubernetes Ingress resources and allows you to manage FortiWeb objects from Kubernetes. It is deployed in a container of a pod in a Kubernetes cluster.

The list below outlines the major functionalities of FortiWeb Ingress Controller:

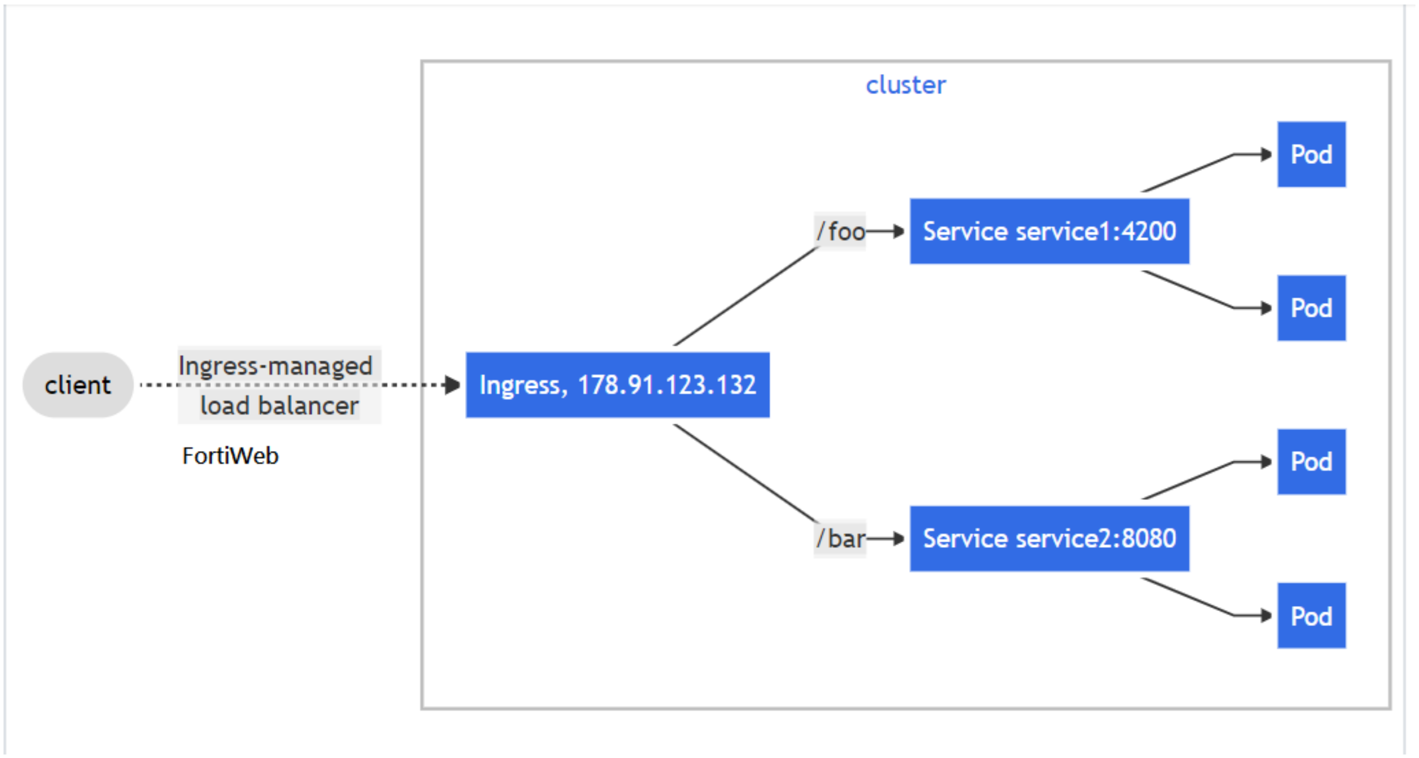
- To list and watch Ingress related resources, such as Ingress, Service, Node and Secret.
- To convert Ingress related resources to FortiWeb objects, such as virtual server, content routing, real server pool, and more.
- To handle Add/Update/Delete events for watched Ingress resources and automatically implement corresponding actions on FortiWeb.

Ingress is a Kubernetes object that manages the external access to services in a Kubernetes cluster (typically HTTP/HTTPS). Ingress may provide load-balancing, SSL termination and name-based virtual hosting.

FortiWeb Ingress Controller combines the capabilities of an Ingress resource with the Ingress-managed Web Application Firewall, FortiWeb.

FortiWeb, as the Ingress-managed Web Application Firewall, protects hosted web applications from attacks that target known and unknown exploits. Using multi-layered and correlated detection methods, FortiWeb defends applications from known vulnerabilities and zero-day threats. The Web Application Security Service from FortiGuard Labs uses information based on the latest application vulnerabilities, bots, suspicious URL and data patterns, and specialized heuristic detection engines to keep your applications safe.

Other features such as health check, traffic log management, and FortiView on FortiWeb facilitates the management of the Kubernetes ingress resources.



Supported Environments

FortiWeb Ingress Controller has been verified to run in the Kubernetes cluster in the below environments:

Environment	Tools for building
Private cloud	kubeadm, minikube, microk8s
Public cloud	Azure AKS, AWS EKS, Oracle OKE

Supported Release and Version

Product	Version
FortiWeb Ingress Controller	1.0.1
Kubernetes	1.19.8 - 1.27.x
FortiWeb Versions	6.3.6 - 7.4.0

The Kubernetes API Version

The Kubernetes API allows you to query and manipulate the state of API objects in Kubernetes (for example, Pods, Nodes, Ingress, and Services).

For the API object, such as the Ingress object you defined in the YAML file, there will be an `apiVersion` field for Kubernetes to determine which API version to deploy the object. Different API versions may have different metadata and specification definitions for that object.

For example, for the Ingress API object, the API versions extensions/v1beta1/Ingress and networking.k8s.io/v1/Ingress would have different data structures for FortiWeb Ingress Controller to parse. Before extensions/v1beta1/Ingress is entirely deprecated by networking.k8s.io/v1/Ingress in Kubernetes v1.22, you may find both API versions are supported by the Kubernetes API server in some cases, such as when upgrading Kubernetes from a lower version v1.16 to a higher version v1.19.

To ensure you use an API version of Kubernetes objects that FortiWeb Ingress Controller supports, you can use the `kubectl` command to check the resource API version.

```
user@control-plane-node:~$ for kind in `kubectl api-resources | tail +2 | awk '{ print
$1 }'`; do kubectl explain $kind; done | grep -e "KIND:" -e "VERSION:"
```

The table below lists the API version of the required API object used for FortiWeb Ingress Controller:

API Object	API Version
Node	v1
Pod	v1
PodTemplate	v1
ServiceAccount	v1
Service	v1
Deployment	apps/v1
ReplicaSet	apps/v1
Event	v1
IngressClass	networking.k8s.io/v1
Ingress	networking.k8s.io/v1
ClusterRoleBinding	rbac.authorization.k8s.io/v1
ClusterRole	rbac.authorization.k8s.io/v1
RoleBinding	rbac.authorization.k8s.io/v1
Role	rbac.authorization.k8s.io/v1

Installation

Install FortiWeb Ingress Controller using Helm Charts.

Currently, only Helm 3 (version 3.6.3 or later) is supported.

Helm Charts ease the installation of FortiWeb Ingress Controller in the Kubernetes cluster. By using the Helm 3 installation tool, most of the Kubernetes objects required for FortiWeb Ingress Controller can be deployed in one simple command.

The Kubernetes objects required for FortiWeb Ingress Controller are listed below:

Kubernetes object	Description
Deployment	By configuring the replica and pod template in the Kubernetes deployment, the deployment ensures FortiWeb Ingress Controller provides a non-terminated service.
Service Account	The service account is used in FortiWeb Ingress Controller.
Cluster Role	A cluster role defines the permission on the Kubernetes cluster-scoped Ingress-related objects.
Cluster Role Binding	The cluster role is bound to the service account used for FortiWeb Ingress Controller, allowing FortiWeb Ingress Controller to access and operate the Kubernetes cluster-scoped Ingress-related objects.
Ingress Class	The IngressClass "fwb-ingress-controller" is created for FortiWeb Ingress Controller to identify the Ingress resource. If the Ingress is defined with the IngressClass "fwb-ingress-controller", FortiWeb Ingress Controller will manage this Ingress resource.

The Helm Chart is composed of a collection of files that describe the related set of Kubernetes required by FortiWeb Ingress Controller; one of which is the `values.yaml` file that provides the default configuration for deploying the Kubernetes objects listed above.

Below lists parts of the value in the `values.yaml` file.

```
image:
  repository: fortinet/fortiweb-ingress
  pullPolicy: IfNotPresent
  tag: "1.0.0"

nameOverride: ""
fullnameOverride: ""

serviceAccount:
  create: true
  annotations: {}
  name: "fortiweb-ingress"

podAnnotations: {}

podSecurityContext: {}

securityContext: {}
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

nodeSelector: {}

tolerations:
  - effect: "NoExecute"
    key: "node.kubernetes.io/not-ready"
    operator: "Exists"
    tolerationSeconds: 30
  - effect: "NoExecute"
    key: "node.kubernetes.io/unreachable"
    operator: "Exists"
    tolerationSeconds: 30

affinity: {}

# Define Ingress Class for FortiWeb Ingress Controller
controller:
  ingressClassResource:
    name: "fwb-ingress-controller"
    enabled: true
    default: true
    controllerValue: "fortinet.com/fwb-ingress-controller"
~
```

In some scenarios, you may want to override some of the values included in the `values.yaml`, such as for an image tag or parameter properties. As the `values.yaml` file is packed in the Helm Chart package, you can override the values when installing or upgrading the Helm Chart (see Install the Helm Chart on page 11 and Upgrade the Helm Chart on page 11). For more details on the parameters, see Configuration parameters on page 23.

To get the verbose output, add `--debug` option for all the Helm commands.

Get Repo Information

To get the repository information:

```
helm repo add FortiWeb-ingress-controller https://fortinet.github.io/fortiweb-ingress/
helm repo update
```

Install the Helm Chart

You can specify a particular Kubernetes namespace in which FortiWeb Ingress Controller will be deployed.

By default, if no Kubernetes namespace is specified, the default namespace would be "default". The `RELEASE_NAME` is the name you give to this chart installation:

```
helm install  [RELEASE_NAME] --namespace [Kubernetes NameSpace] \
FortiWeb-ingress-controller/fwb-k8s-ctrl
```

In the example below, the Helm chart is installed with the release name "first-release" in the Kubernetes namespace "FortiWeb-ingress":

```
user@control-plane-node ~> helm install first-release --namespace FortiWeb-ingress \
FortiWeb-ingress-controller/fwb-k8s-ctrl
```

If you want to override values in the Helm Chart, you can use `--set` flags in the command. In the example below, you can set the virtualServerWafProfile parameter as mandatory:

```
user@control-plane-node ~> helm install --debug first-release \
--set parameters.virtualServerWafProfile="mandatory" \
--namespace FortiWeb-ingress FortiWeb-ingress-controller/fwb-k8s-ctrl
```

Moreover, you can create a new namespace and deploy FortiWeb Ingress Controller within the namespace at the same time:

```
helm install first-release --namespace FortiWeb-ingress \
--create-namespace --wait FortiWeb-ingress-controller/fwb-k8s-ctrl
```

Upgrade the Helm Chart

You can specify the namespace with the `--namespace` option. Use `--install` option to install the release with RELEASE_NAME if it does not exist.

Note: The `--reset-values` option will remove all the user-supplied values. For example, if you had specified the virtualServerWafProfile parameter to be mandatory in a previous upgrade or install, the value will be reset to optional. The `--reset-values` option ensures all the values are directly from the updated repository.

```
helm repo update
helm upgrade --reset-values --debug -n [Kubernetes NameSpace] [RELEASE_NAME] \
FortiWeb-ingress-controller/fwb-k8s-ctrl --install
```

You can also change the field of `values.yaml` with the `--set` command.

To see which values you can change, please refer to https://github.com/fortinet/fortiweb-ingress/blob/main/charts/fwb-k8s-ctrl-1.0.0/values.yaml.

Using the `--debug` option, you can check the Helm debug information “USER-SUPPLIED VALUES” to check if you have all the value set as you need.

```
Release "first-release" has been upgraded. Happy Helming!
NAME: first-release
LAST DEPLOYED: Mon Apr 18 09:07:46 2022
NAMESPACE: FortiWeb-ingress
STATUS: deployed
REVISION: 2
TEST SUITE: None
USER-SUPPLIED VALUES:
parameters:
  virtualServerWafProfile: mandatory
```

Check the Installation

Check to see if the FortiWeb Ingress Controller is installed correctly:

```
helm history -n [Kubernetes NameSpace] [RELEASE_NAME]
```

The helm history command shows the installation information:

```
user@control-plane-node ~> helm history -n FortiWeb-ingress first-release
REVISION        UPDATED                         STATUS          CHART
  APP VERSION     DESCRIPTION
1               Tue Feb  8 05:37:33 2022        superseded      fwb-k8s-ctrl-0.1.0
 1.0.0           Install complete
```

You can also use the kubectl command to check the installation:

```
kubectl get -n [namespace] deployments
```

```
kubectl get -n [namespace] pods
```

You will get the FortiWeb Ingress Controller deployment and pod status like the following:

```
user@control-plane-node ~> kubectl get -n FortiWeb-ingress deployments
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
first-release-fwb-k8s-ctrl   1/1     1            1           8s
```

```
user@control-plane-node ~> kubectl get -n FortiWeb-ingress pods
NAME                                              READY   STATUS    RESTARTS   AGE
first-release-fwb-k8s-ctrl-6447856856-h5skx   1/1     Running   0          8s
```

Check the log of the FortiWeb Ingress Controller:

```
kubectl logs -n [namespace] -f [pod name]
```

You can get the FortiWeb Ingress Controller logs like the following:

```
user@control-plane-node ~> kubectl logs -n FortiWeb-ingress -f \
first-release-fwb-k8s-ctrl-6447856856-h5skx

Starting FortiWeb ingress controller
time=="2021-10-13T06:27:56Z"level=info msg="Starting FortiWeb Ingress controller"
```

Uninstall the Helm Chart

To uninstall the Helm Chart:

```
helm uninstall [RELEASE_NAME]
```

To uninstall the FortiWeb Ingress Controller in the specified Kubernetes namespace:

```
helm uninstall [RELEASE_NAME] --namespace [Kubernetes NameSpace]
```

Kubernetes Ingress

Kubernetes Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

An Ingress can be configured to give Kubernetes services externally reachable URLs, load balance traffic, terminate SSL/TLS, and offer name-based virtual hosting.

It is important to note that to satisfy an Ingress, you must have an Ingress controller (in the example above, FortiWeb Ingress Controller is used). Only creating an Ingress resource would have no effect.

FortiWeb Ingress Controller version 1.0.0 only supports services of type **NodePort**.

Ingress class

Ingresses can be implemented by different controllers, often with different configurations. Each Ingress should specify an IngressClass name, which is a reference to an IngressClass resource that contains additional configuration including the name of the controller that should implement the class.

For an Ingress that would be implemented by FortiWeb Ingress Controller, please specify the `ingressClassName` to `fwb-ingress-controller` in the ingress specification. Upon installation, FortiWeb Ingress Controller is set as the default Ingress controller in the Helm chart value.yaml.

```
controller:
  ingressClassResource:
    name: "fwb-ingress-controller"
    enabled: true
    default: true
    controllerValue: "fortinet.com/fwb-ingress-controller"
```

Mapping of the Ingress related resources with the FortiWeb objects

Kubernetes Objects	FortiWeb Objects
Ingress	Virtual server Content Routing Scripting
Service	Real Server Pool

Kubernetes Objects	FortiWeb Objects
	Real Server
Node	Real Server
Secret	Local Certificate Local Certificate Group Client-SSL

Naming rule

For FortiWeb objects created by FortiWeb Ingress Controller, the name of the object is composed of the namespace and the name of the Kubernetes objects. The naming rule is shown below:

FortiWeb Objects	Naming Rule
Virtual server Real Server Pool Scripting Local Certificate Local Certificate Group Client-SSL	[namespace of Kubernetes objects]_[name of Kubernetes objects]
Content Routing	[namespace of Kubernetes objects]_[name of Kubernetes ingress]_[name of Kubernetes service]
Real Server	Name of the Kubernetes node

Ingress types

FortiWeb supports all 5 types of Ingress:

1. Default backend on page 16
2. Minimal-Ingress on page 17
3. Name-based virtual hosting on page 17
4. Hostname wildcards on page 19
5. TLS on page 20

For details on each Ingress type, see https://kubernetes.io/docs/concepts/services-networking/ingress/.

For an example Ingress file, see https://github.com/fortinet/fortiweb-ingress/tree/main/service_examples/.

Ensure the service used in the Ingress is already deployed with **NodePort** type. Kubernetes does not verify whether the service defined in the Ingress exists or not when deploying an Ingress. So, if you remove the service used in the deployed Ingress, you will not be warned or blocked from this action.

Default backend

An Ingress with no rules sends all traffic to a single default backend. If none of the hosts or paths match the HTTP request in the Ingress objects, the traffic is routed to your default backend. Therefore, if Rules are not specified, `defaultBackend` must be specified in the Ingress.

Note: FortiWeb Ingress Controller only supports `Service` backend. A `Resource` backend is not supported.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the Default backend Ingress YAML example, follow this link: https://github.com/fortinet/fortiweb-ingress/blob/main/ingress_examples/default_backend.yaml.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-with-default-backend
  annotations: {
    "fortiweb-ip" : "172.23.133.148",
    "fortiweb-login" : "fwb-login1",
    "fortiweb-port": "443",
    "fortiweb-ctrl-log" : "enable",
    "virtual-server-ip" : "192.23.133.6",
    "virtual-server-addr-type" : "ipv4",
    "virtual-server-interface" : "port2",
    "server-policy-web-protection-profile" : "Inline Standard Protection",
    "server-policy-https-service" : "HTTPS",
    "server-policy-http-service" : "HTTP",
    "server-policy-syn-cookie" : "enable",
    "server-policy-http-to-https" : "disable"
  }

spec:
  ingressClassName: fwb-ingress-controller
  defaultBackend:
    service:
      name: service1
      port:
        number: 1241
```

You can add defaultBackend in any particular Ingress. You can check the example here:

https://github.com/fortinet/fortiweb-ingress/blob/main/service_examples/default-http-backend.yaml

Minimal-Ingress

A minimal-Ingress has at least one rule defined in the Ingress with no specified default backend. The following is an example of a minimal-Ingress.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the Minimal Ingress YAML example, follow this link: https://raw.githubusercontent.com/fortinet/fortiweb-ingress/main/ingress_examples/.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations: {
    "fortiweb-ip" : "172.23.133.148",
    "fortiweb-login" : "fwb-login1",
    "fortiweb-ctrl-log" : "disable",
    "virtual-server-ip" : "192.23.133.6",
    "virtual-server-addr-type" : "ipv4",
    "virtual-server-interface" : "port2",
    "server-policy-web-protection-profile" : "Inline Standard Protection",
    "server-policy-https-service" : "HTTPS",
    "server-policy-http-service" : "HTTP",
    "server-policy-syn-cookie" : "enable",
    "server-policy-http-to-https" : "disable"
  }
spec:
  ingressClassName: fwb-ingress-controller
  rules:
  - host: test.com
    http:
      paths:
      - path: /info
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 1241
```

Name-based virtual hosting

Name-based virtual hosts support the routing of HTTP/HTTPS traffic to multiple host names at the same IP address.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the Name-based virtual hosting Ingress YAML example, follow this link: https://github.com/fortinet/fortiweb-ingress/blob/main/ingress_examples/simple-fanout-example.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations: {
    "fortiweb-ip" : "172.23.133.148",
    "fortiweb-login" : "fwb-login1",
    "fortiweb-ctrl-log" : "disable",
    "virtual-server-ip" : "192.23.133.6",
    "virtual-server-addr-type" : "ipv4",
    "virtual-server-interface" : "port2",
    "server-policy-web-protection-profile" : "Inline Standard Protection",
    "server-policy-https-service" : "HTTPS",
    "server-policy-http-service" : "HTTP",
    "server-policy-syn-cookie" : "enable",
    "server-policy-http-to-https" : "disable"
  }
spec:
  ingressClassName: fwb-ingress-controller
  rules:
  - host: test.com
    http:
      paths:
      - path: /info
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 1241
      - path: /hello
        pathType: Prefix
        backend:
          service:
            name: service2
            port:
              number: 1242
```

Hostname wildcards

The hostname can be a precise match or a wildcard. Precise matches require the HTTP host header to match the host field (e.g., `foo.bar.com`). Wildcard matches require the HTTP host header to be equal to the suffix of the wildcard rule (e.g., `*.foo.com`).

Refer to the examples below to determine whether an HTTP host header is a wildcard match or not.

Host	Host header	Does it match?
`*.foo.com`	`bar.foo.com`	Yes, it matches based on shared suffix.
`*.foo.com`	`baz.bar.foo.com`	No, it does not match. Wildcard only covers a single DNS label.
`*.foo.com`	`foo.com`	No, it does not match. Wildcard only covers a single DNS label.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the Hostname wildcards Ingress YAML example, follow this link: https://github.com/fortinet/fortiweb-ingress/blob/main/ingress_examples/ingress-wildcard-host.yaml.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-wildcard-host
  annotations: {
    "fortiweb-ip" : "172.23.133.148",
    "fortiweb-login" : "fwb-login1",
    "fortiweb-ctrl-log" : "disable",
    "virtual-server-ip" : "192.23.133.6",
    "virtual-server-addr-type" : "ipv4",
    "virtual-server-interface" : "port2",
    "server-policy-web-protection-profile" : "Inline Standard Protection",
    "server-policy-https-service" : "HTTPS",
    "server-policy-http-service" : "HTTP",
    "server-policy-syn-cookie" : "enable",
    "server-policy-http-to-https" : "disable"
  }

spec:
  ingressClassName: fwb-ingress-controller
  rules:
  - host: "test.com"
    http:
      paths:
```

```
      - path: /info
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 1241
  - host: "*.test1.com"
    http:
      paths:
      - path: /info
        pathType: Prefix
        backend:
          service:
            name: service2
            port:
              number: 1242
```

TLS

You can secure an Ingress by specifying a Secret that contains a TLS private key and certificate. The Ingress resource only supports a single TLS port, 443, and assumes TLS termination at the Ingress point (traffic to the Service and its Pods is in plain text). If the TLS configuration section in an Ingress specifies different hosts, they are multiplexed on the same port according to the hostname specified through the SNI TLS extension (provided the Ingress controller supports SNI). The TLS secret must contain keys named `tls.crt` and `tls.key` that contain the certificate and private key to use for TLS.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the TLS key YAML example, follow this link:
https://kubernetes.io/docs/concepts/services-networking/ingress/#tls

For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: testsecret-tls
  namespace: default
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
type: kubernetes.io/tls
```

You can create the TLS secret by using the kubectl command, for example:

```
kubectl create secret tls  tls-example --cert=/etc/ssl/tls.crt --key=/etc/ssl/tls.key
```

Referencing this secret in an Ingress tells the Ingress controller to secure the channel from the client to the load balancer using TLS. You need to ensure the TLS secret you created came from a certificate that contains a Common Name (CN), also known as a Fully Qualified Domain Name (FQDN) for `https-example.foo.com`.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the TLS Ingress YAML example, follow this link: https://github.com/fortinet/fortiweb-ingress/blob/main/ingress_examples/tls-example-ingress.yaml.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tls-example-ingress
  annotations: {
    "fortiweb-ip" : "172.23.133.148",
    "fortiweb-login" : "fwb-login1",
    "fortiweb-ctrl-log" : "disable",
    "virtual-server-ip" : "192.23.133.6",
    "virtual-server-addr-type" : "ipv4",
    "virtual-server-interface" : "port2",
    "server-policy-web-protection-profile" : "Inline Standard Protection",
    "server-policy-https-service" : "HTTPS",
    "server-policy-http-service" : "HTTP",
    "server-policy-syn-cookie" : "enable",
    "server-policy-http-to-https" : "disable"
  }

spec:
  ingressClassName: fwb-ingress-controller
  tls:
  - hosts:
    - test.com
    secretName: tls-example
  rules:
  - host: test.com
    http:
      paths:
      - path: /info
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 1241
      - path: /hello
        pathType: Prefix
```

```
backend:
  service:
    name: service2
    port:
      number: 1242
```

Configuration parameters

FortiWeb Authentication Secret

FortiWeb Ingress Controller satisfies an Ingress by FortiWeb REST API call, so the authentication parameters of FortiWeb must be known to FortiWeb Ingress Controller.

To preserve the authentication securely on the Kubernetes cluster, you can save it with the Kubernetes secret.

For example:

```
kubectl create secret generic fwb-login1 -n [namespace] \ --from-lit-
eral=username=admin --from-literal=password=[admin password]
```

The secret is named `fwb-login`. This value will be specified in the Ingress annotation "FortiWeb-login" for FortiWeb Ingress Controller to get permission access on FortiWeb.

The namespace of the authentication secret must be the same as the Ingress which references this authentication secret.

Annotation in Ingress

Configuration parameters are required to be specified in the Ingress annotation to enable FortiWeb Ingress Controller to determine how to deploy the Ingress resource.

Parameter	Description
fortiweb-ip	The IP address to log in to FortiWeb.
fortiweb-login	The Kubernetes secret name preserves the FortiWeb authentication information.
fortiweb-port	The network port to log in to Fortiweb.
fortiweb-ctrl-log	The virtual server IP of the virtual server to be configured on FortiWeb. This IP will be used as the address of the Ingress Controller.
machine-learning-anomaly-add-domain	The domain list added to Anomaly detection. For more information, see "ML Based Anomaly Detection" in FortiWeb Administration Guide.
machine-learning-anomaly-add-ip	The IP list added to Anomaly detection. For more information, see "ML Based Anomaly Detection" in FortiWeb Administration Guide.
machine-learning-anomaly-ip-list-type	The IP address should block or accept.

Parameter	Description
	For more information, see "ML Based Anomaly Detection" in FortiWeb Administration Guide.
machine-learning-anomaly-del-domain-list	The domain list deleted from Anomaly detection. For more information, see "ML Based Anomaly Detection" in FortiWeb Administration Guide.
machine-learning-anomaly-del-ip	The IP list deleted from Anomaly detection. For more information, see "ML Based Anomaly Detection" in FortiWeb Administration Guide.
machine-learning-bot-add-ip	The IP list added to ML Based Bot Detection. For more information, see "Configuring ML Based Bot Detection policy" in FortiWeb Administration Guide.
machine-learning-bot-del-ip	The IP list deleted from ML Based Bot Detection. For more information, see "Configuring ML Based Bot Detection policy" in FortiWeb Administration Guide.
machine-learning-api-add-domain	The domain list added to ML Based API Protection. For more information, see "Configuring ML Based API Protection policy" in FortiWeb Administration Guide.
machine-learning-api-del-domain	The domain list deleted from ML Based API Protection. For more information, see "Configuring ML Based API Protection policy" in FortiWeb Administration Guide.
machine-learning-api-add-ip	The IP list added to ML Based API Protection. For more information, see "Configuring ML Based API Protection policy" in FortiWeb Administration Guide.
machine-learning-api-del-ip	The IP list deleted from ML Based API Protection. For more information, see "Configuring ML Based API Protection policy" in FortiWeb Administration Guide.
machine-learning-api-ip-list-type	The IP address should block or accept in ML Based API Protection. For more information, see "Configuring ML Based API Protection policy" in FortiWeb Administration Guide.
virtual-server-addr-type	IPv4 or IPv6.
virtual-server-ip	The virtual server IP of the virtual server to be configured on FortiWeb. This IP will be used as the address of the Ingress.
virtual-server-interface	The FortiWeb network interface for the client to access the virtual server.
server-policy-web-protection-profile	The name of the web protection profile to be applied to the virtual server.
server-policy-https-service	The HTTPS service name.
server-policy-http-service	The HTTP service name.

Parameter	Description
server-policy-syn-cookie	Enable/Disable the SYN cookie.
server-policy-http-to-https	Enable/Disable the http-to-https funciton.

For more details on configuring parameters with virtual-server prefix and load-balance prefix, please reference FortiWeb Handbook on Configuring virtual servers.

Annotation in Service

You can define the health check profile and SSL profile in the Kubernetes service annotation.

The health check profile and SSL profile will be automatically configured in the corresponding real server pool on FortiWeb.

Parameter	Description
health-check-ctrl	Enable/disable the health checking for the back-end server pool. For more information, see "Defining your web servers" in FortiWeb Administration Guide.
lb-algo	Specify the load balancing algorithm. For more information, see "Defining your web servers" in FortiWeb Administration Guide.
persistence	Specify the persistence rule. For more information, see "Defining your web servers" in FortiWeb Administration Guide.

Deployment

Deploy the Pods and expose the Services

```
Service1:

kubectl apply -f https://raw.githubusercontent.com/fortinet/fortiweb-ingress/-
main/service_examples/service1.yaml

Service2:

kubectl apply -f https://raw.githubusercontent.com/fortinet/fortiweb-ingress/-
main/service_examples/service2.yaml
```

Check the service1 and service2 you have deployed.

```
kubectl get service

NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
service1               NodePort    10.111.143.250   <none>         1241:31320/TCP   10m

service2               NodePort    10.109.117.79    <none>         1242:32075/TCP
2m59s
```

Deploy the Ingress

Define the Simple-fanout Ingress resource.

Due to PDF formatting limitations, the code example below would not retain indentations if copy and pasted directly into a YAML file. Without the proper indentations, the YAML will be invalid.

To copy the Simple-fanout Ingress YAML example, follow this link: https://github.com/fortinet/fortiweb-ingress/blob/main/ingress_examples/simple-fanout-example.yaml.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations: {
    "fortiweb-ip" : "172.23.133.148",
    "fortiweb-login" : "fwb-login1",
    "fortiweb-ctrl-log" : "disable",
    "virtual-server-ip" : "192.23.133.6",
    "virtual-server-addr-type" : "ipv4",
    "virtual-server-interface" : "port2",
    "server-policy-web-protection-profile" : "Inline Standard Protection",
    "server-policy-https-service" : "HTTPS",
    "server-policy-http-service" : "HTTP",
    "server-policy-syn-cookie" : "enable",
    "server-policy-http-to-https" : "disable"
  }
spec:
  ingressClassName: fwb-ingress-controller
  rules:
  - host: test.com
    http:
      paths:
      - path: /info
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 1241
      - path: /hello
        pathType: Prefix
        backend:
          service:
            name: service2
            port:
              number: 1242
```

Deploy it with `kubectl` command.

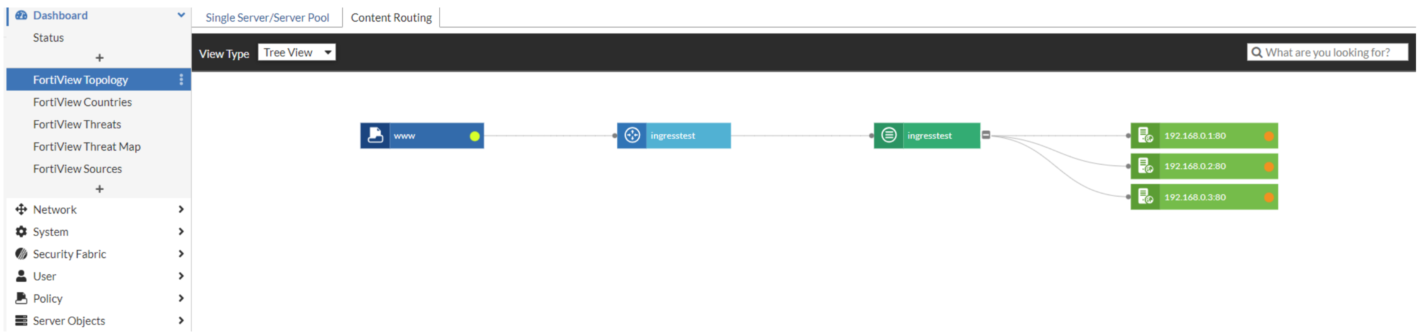
```
kubectl apply -f simple-fanout.yaml
ingress.networking.k8s.io/simple-fanout-example created
```

Get the information of the simple-fanout-example Ingress by using the kubectl describe command.

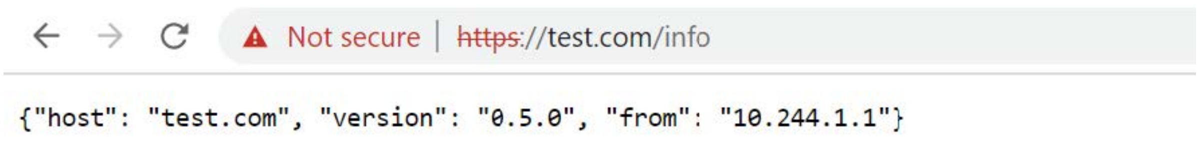
```
user@control-plane-node ~> kubectl describe ingress simple-fanout-example

Name:             simple-fanout-example


Namespace:        default

Address:          192.23.133.6


Default backend:  default-http-backend:80


Rules:
  Host        Path  Backends

  ----        ----  --------
  test.com
              /info    service1:1241 (10.244.1.16:9876)
              /hello   service2:1242 (10.244.12.26:80)

Annotations:   FortiWeb-admin: admin
              "fortiweb-ip" : "172.23.133.148",
              "fortiweb-login" : "fwb-login1",
              "fortiweb-ctrl-log" : "disable",
              "virtual-server-ip" : "192.23.133.6",
              "virtual-server-addr-type" : "ipv4",
              "virtual-server-interface" : "port2",
              "server-policy-web-protection-profile" : "Inline Standard Protection",
              "server-policy-https-service" : "HTTPS",
              "server-policy-http-service" : "HTTP",
              "server-policy-syn-cookie" : "enable",
              "server-policy-http-to-https" : "disable"
```

FortiView

Check the deployed Ingress with FortiView.

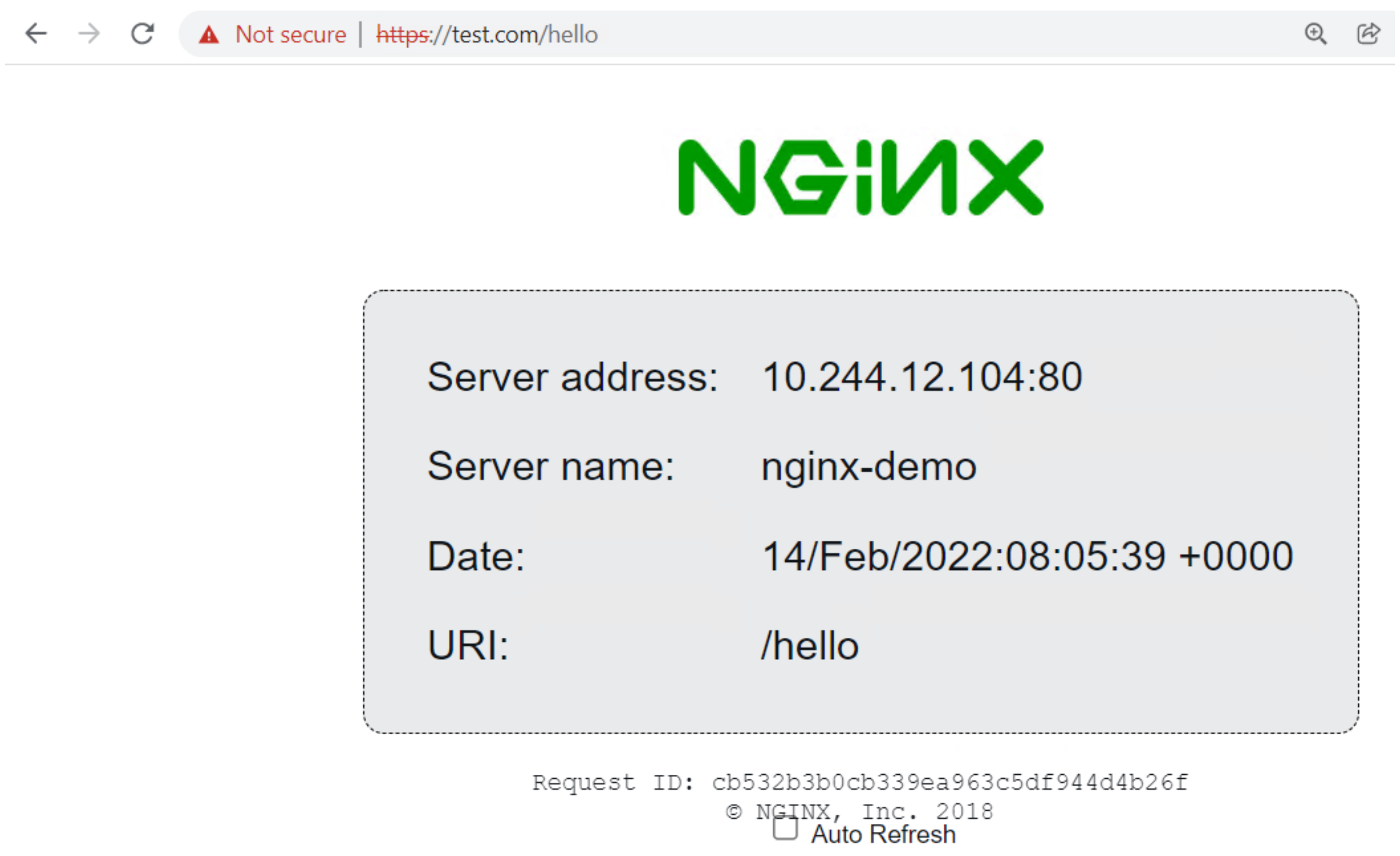


Try to access `https://test.com/info`.



```
{"host": "test.com", "version": "0.5.0", "from": "10.244.1.1"}
```

Try to access `https://test.com/hello`.



Server address: 10.244.12.104:80

Server name: nginx-demo

Date: 14/Feb/2022:08:05:39 +0000

URI: /hello

Request ID: cb532b3b0cb339ea963c5df944d4b26f
© NGINX, Inc. 2018

Update or delete the Ingress

To update an Ingress resource:

You can edit the `ingress.yaml.` and use `kubectl apply` or use the `kubectl edit` command.

```
kubectl edit ingress simple-fanout-example
```

To delete the Ingress resource:

```
kubectl delete ingress/simple-fanout-example
```

Add, update or delete Service and Node

Service

FortiWeb Ingress Controller only monitors **port sections** and **annotations** defined in services used in the deployed Ingress resource. For example, let the service2 also handle traffic with the TCP destination port 8080 to the nginx pod. Use the `kubectl edit` command to see the original service2 spec.

```
kubectl edit service service2
```

```
#original definition of service2
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2021-10-21T08:50:31Z"
  labels:
    run: nginx-demo
  name: service2
  namespace: default
  resourceVersion: "26766217"
  selfLink: /api/v1/namespaces/default/services/service2
  uid: 69aa596e-1f23-4696-b770-6202654058a5
spec:
  clusterIP: 10.109.117.79
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 32075
    port: 1242
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx-demo
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Now, add another port and give each port a name. In the example below, take note of the code in bold text.

```
# Modified service2
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2021-10-21T08:50:31Z"
  labels:
    run: nginx-demo
  name: service2
  namespace: default
  resourceVersion: "26766217"
  selfLink: /api/v1/namespaces/default/services/service2
  uid: 69aa596e-1f23-4696-b770-6202654058a5
spec:
  clusterIP: 10.109.117.79
  externalTrafficPolicy: Cluster
  ports:
  - name: http-80
    nodePort: 32075
    port: 1242
    protocol: TCP
    targetPort: 80
  - name: http-8080
    port: 1243
    protocol: TCP
    targetPort: 8080
  selector:
    run: nginx-demo
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Check the service with the `kubectl get` command. You can see service2 has registered with the second port 1243 and get a NodePort 31879 allocated from Kubernetes.

```
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
           AGE

service1                NodePort    10.111.143.250   <none>        1241:31320/TCP
           4d21h

service2                NodePort    10.109.117.79    <none>
1242:32075/TCP,1243:31879/TCP   4d21h
```

And check the FortiWeb real server pool default_service2. You can see the pool members with port 31879 are added.

Note: If you delete the service used in the deployed Ingress resources, Kubernetes would not give you any warning, and FortiWeb Ingress Controller would not handle any delete events on the service.

Node

If you add or delete a worker node, FortiWeb Ingress Controller will check the deployed Ingress resources and handle the add/delete event. For updating a node, FortiWeb Ingress Controller only monitors the node's IP.

Debug

By default, the FortiWeb Ingress Controller records the process of the Ingress implementation in verbose mode. The debug log shows the check of annotation parameters and the process of calling the REST API to FortiWeb when deploying, updating, or deleting the Ingress resources.

The verbose log can be enabled or disabled for any particular Ingress. This is determined by the status of the FortiWeb-ctrl-log configuration parameter. For more information, see Configuration parameters on page 23.

To see the log, you can use the `kubectl logs` command:

```
kubectl logs -n [namespace] -f [FortiWeb Ingress Controller pod name]
```

The log shows which problem you encounter.

Some troubleshooting steps may require restarting the FortiWeb Ingress Controller. For example, the FortiWeb Ingress Controller may not connect to FortiWeb after changing the network firewall rule. To fix this type of environment issue, you can restart the FortiWeb Ingress Controller by using the following command:

```
kubectl -n [namespace] rollout restart deployment/[ FortiWeb Ingress Controller deploy-
ment name]
```

FAQ

1) What should I do if I find a Kubernetes API object is supported by multiple API groups?

You may encounter this scenario when upgrading your Kubernetes cluster and the higher Kubernetes version has deprecated some of the API groups.

For example, extensions/v1beta1/Ingress is entirely deprecated by networking.k8s.io/v1/Ingress in Kubernetes v1.22. You may find extensions/v1beta1/Ingress and networking.k8s.io/v1/Ingress both exist in your system when upgrading the Kubernetes cluster from v1.16 to v1.20.

In this case, you have to disable the API group extensions/v1beta1 to ensure the system use the API group networking.k8s.io/v1 that is supported by FortiWeb Ingress Controller. After disabling the API group, the Kubernetes API server needs to be restarted to apply the changes.

Follow the steps below:

1. Edit `/etc/kubernetes/manifests/kube-apiserver.yaml`
2. Under spec.containers.command, add `--runtime-config=extensions/v1beta1=false`
3. Restart the Kubernetes API Server using `systemctl restart kubelet.service`

For more information on enabling/disabling deprecated API groups, see https://kubernetes.io/docs/reference/using-api/#enabling-or-disabling.

FORTINET®

www.fortinet.com