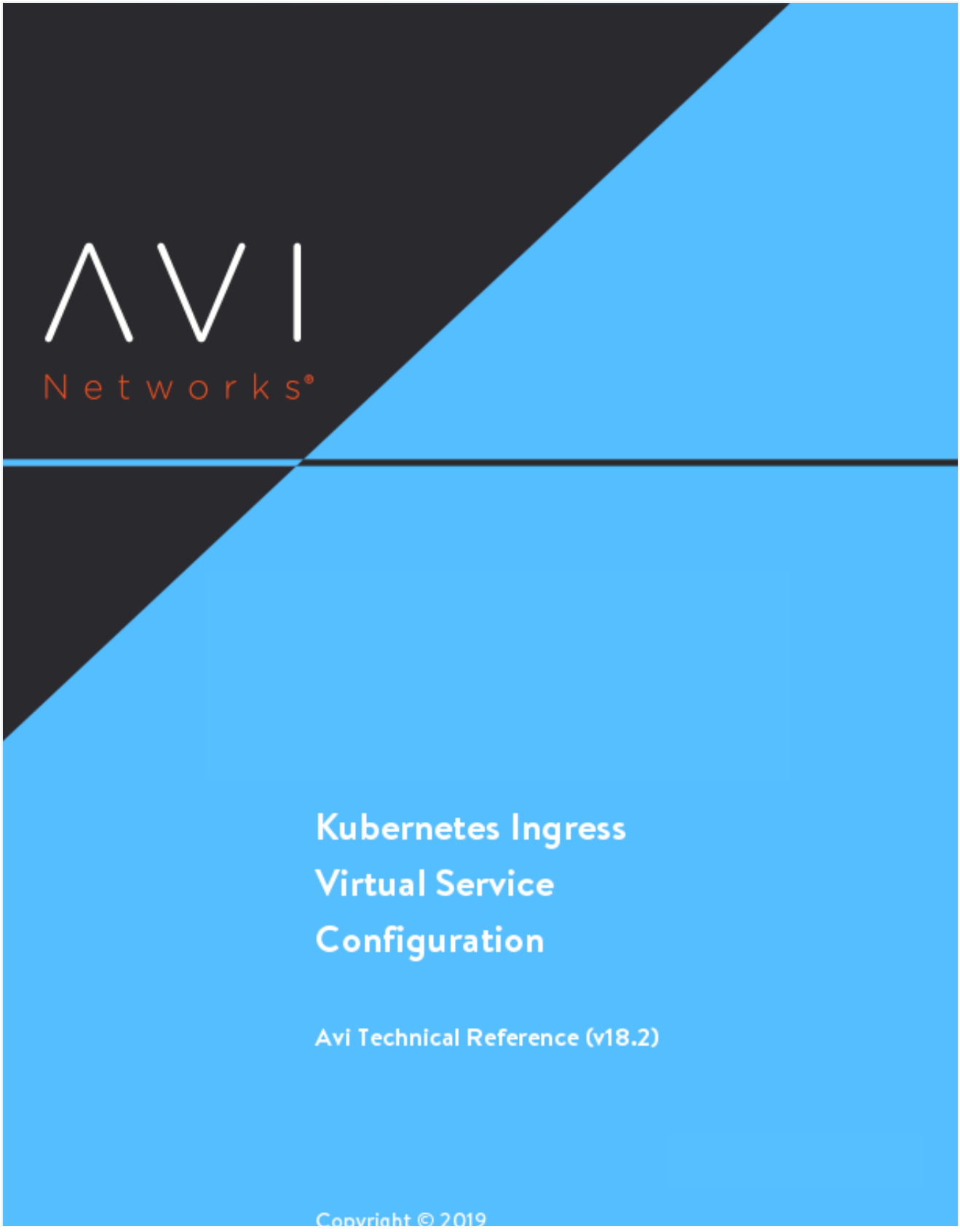
AVI
Networks®
Kubernetes Ingress
Virtual Service
Configuration
Avi Technical Reference (v18.2)
Copyright © 2019

Kubernetes Ingress Virtual Service Configuration

view online

In a Kubernetes cloud, Kubernetes Ingress creation triggers creation of north-south or Ingress Avi Vantage virtual service and pool objects. OpenShift/Kubernetes Service Configuration in Avi Vantage explains how Kubernetes services map to Avi Vantage's virtual service and pool objects. This article explains how Kubernetes Ingresses trigger virtual service and pool object creation in Avi Vantage.

Kubernetes Ingresses to Avi Vantage Object Mapping

Kubernetes ingresses can be configured in one of two ways as far as virtual IPs are concerned.

- Dedicated virtual IP per ingress: Each Kubernetes ingress is allocated its own virtual service/pool objects and a virtual IP (VIP). This provides the best performance, availability and isolation, but consumes an IP address per route.
- Shared virtual service across multiple ingresses: Each Kubernetes ingress is tied to a pre-created parent shared virtual service and will use the shared virtual service?s virtual IP. This conserves IP addresses, but provides variable performance, no isolation, and shares fate with all sibling routes. A parent shared virtual service is pre-created with HTTP/HTTPS application profiles with listeners on ports 80 and 443 for every Service Engine group configured in the cloud. The parent virtual service's name is of the form `parent-vs-SEGroupname-Cloudname`.

The cloud configuration `override_service_ports` has been introduced. By default, this knob is disabled. When the knob is not set, virtual service for Ingress will use ports from Kubernetes service. When the knob is set, virtual services for Ingress will use well known ports i.e. 80 for HTTP and 443 for HTTPS.

Note: Support for shared virtual services is available beginning 17.2.3 in the 17.2 release series.

The following table maps each type of Kubernetes ingress to Avi Vantage objects.

Kubernetes ingress	Avi Vantage Object	Comment
Dedicated virtual service per ingress	VirtualService, Pool	Every ingress creates a VirtualService, Pool object. Ingress annotation *avi_proxy: '{"dedicated_route": true}'* creates a dedicated virtual service (that uses a VIP) per ingress in shared VirtualService Projects/Namespaces. Layer4 (TCP, UDP) ingresses require a dedicated VIP per ingress
Shared virtual service: HTTP Virtual Hosting ingresses	Pool	HTTP policy is used in parent virtual service to switch host/path to the appropriate pool.
Shared virtual service: HTTPS ingresses	Child virtual hosted/SNI VirtualService, Pool	SNI is used to route traffic to the appropriate child VirtualService/Pool.

Shared Virtual Service Mode Configuration

- The field `shared_virtualservice_namespace` configures the cloud to operate in either dedicated or shared virtual service mode. Enable Use Shared Virtual Service in the UI, if required. Enabling `shared_virtualservice_namespace` uses the shared parent virtual service for all ingresses in all namespaces by

default. Disabling `shared_virtualservice_namespace` creates a dedicated virtual service per ingress in all namespaces by default.

- Changing this mode is disruptive and can be time consuming with a large number of ingresses; hence, it is recommended that this be configured once at initial cloud configuration time.
- Individual namespaces can be overridden to operate in either shared or dedicated modes using the `avi_virtualservice: shared|dedicated` namespace annotation.
  - If `shared_virtualservice_namespace` is enabled for the cloud, namespace annotation `avi_virtualservice: dedicated` creates dedicated virtual services for all ingresses in the namespace.
  - On the contrary, if `shared_virtualservice_namespace` is disabled for the cloud, namespace annotation `avi_virtualservice: shared` uses the shared virtual service for all ingresses in the namespace.
- Every ingress can be overridden to create a dedicated virtual service for itself, even if the cloud or the namespace is operating in the shared mode. Ingress annotation `avi_proxy: '{"dedicated_route": true}'` creates a dedicated virtual service for that Ingress. The most common use case for dedicated virtual service is for non-HTTP(S) layer 4 (TCP, UDP) Ingresses.

Ingress Configuration Examples

The following examples assume the following cloud configuration.

- `http_container_ports` is configured with 80, 8080.
- A network object is created with an IP address pool for north-south VIPs; a north-south IPAM profile is created and linked to the network object and the cloud is linked to the north-south IPAM profile.
- A network object is created with an IP address pool for east-west VIPs; an east-west IPAM profile is created and linked to the network object and the cloud is linked to the east-west IPAM profile.

Sample Deployment and Service

The following deployment and service are examples for deploying HTTP pods.

Sample deployment configuration in YAML format.

```
kind: Deployment
apiVersion: apps/v1beta2
metadata:
  name: avitest-deployment
  labels:
    app: avitest
spec:
  replicas: 2
  selector:
    matchLabels:
      app: avitest
  template:
    metadata:
      labels:
        app: avitest
    spec:
      containers:
      - name: avitest
        image: avinetworks/server-os
        ports:
```

```
        - name: http
          containerPort: 8080
          protocol: TCP
```

Sample service file to create an east-west service in YAML format.

```
kind: Service
apiVersion: v1
metadata:
  name: avisvc
  labels:
    svc: avisvc
spec:
  ports:
  - name: http
    port: 80
    targetPort: 8080
  selector:
    app: avitest
```

Since `default_service_as_east_west_service` is enabled in the cloud configuration, this service creates an east-west virtual service.

If HTTPS pods are deployed, container port is usually 443 or 8443 and service port is 443 or 8443.

Sample Ingresses

Dedicated VIP for HTTP Ingress

In this case `shared_virtualservice_namespace` is disabled in the cloud and namespace does not have the `avi_virtualservice: shared` annotation.

The following sample ingress file creates a http ingress with a dedicated VIP for service avisvc.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: aviingress
spec:
  rules:
  - host: aviingress1.default.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc1
          servicePort: 80
        path: /foo
      - backend:
          serviceName: avisvc2
```

```
          servicePort: 80
        path: /bar
  - host: aviingress2.default.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc3
          servicePort: 80
```

This dedicated virtual service ingress will create its own virtual service and pool with its dedicated VIP. Virtual service name is of the form `ingressname.dns_sub_domain`. In the example above, a virtual service called `aviingress.default.acme.local` will be created.

Shared VIP for HTTP Ingress

In this case either `shared_virtualservice_namespace` is enabled in the cloud or namespace has the `avi_virtualservice: shared` annotation.

The following sample ingress file creates a HTTP route associated with parent virtual service `parent-vs-Default-Group-Default-Cloud` for service `avisvc`.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: aviingress
spec:
  rules:
  - host: aviingress1.default.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc1
          servicePort: 80
        path: /foo
      - backend:
          serviceName: avisvc2
          servicePort: 80
        path: /bar
  - host: aviingress2.default.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc3
          servicePort: 80
```

The HTTP application will be created as a pool group and pool for the parent virtual service. Pool name will be of the form `namespace-host--path-aviroute-pool-port-protocol`. For example, ingress with host `aviingress1.default.`

`acme.local` and path `/foo` in namespace `default` will have a pool name `default-aviingress1.default.acme.local--foo-aviroute-pool-http-tcp`. For the HTTP ingress object above, three Pools will be created for the three Services. An HTTP policy will provide host/path switching for the pool.

Dedicated VIP for HTTPS with Edge Termination

In this case `shared_virtualservice_namespace` is enabled in the cloud but namespace has the `avi_virtualservice: dedicated` annotation.

The `avi_virtualservice: dedicated` namespace annotation overrides the `shared_virtualservice_namespace` cloud configuration and creates dedicated virtual services for all routes in the namespace.

Namespace looks as follows.

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    avi_virtualservice: dedicated
  name: ns1
spec:
  finalizers:
  - kubernetes
status:
  phase: Active
```

The following sample Ingress file creates a HTTPS ingress with edge termination with a dedicated VIP.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: aviingress
spec:
  rules:
  - host: aviingress1.ns1.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc1
          servicePort: 80
        path: /foo
  - host: aviingress2.ns1.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc2
          servicePort: 80
  tls:
    - secretName: aviingress1cert
```

```
      hosts:
      - aviingress1.ns1.acme.local
    - secretName: aviingress2cert
      hosts:
      - aviingress2.ns1.acme.local
  backend:
    serviceName: avisvc3
    servicePort: 80
```

This dedicated virtual service ingress will create its own virtual service with its dedicated VIP. Virtual service name is of the form `ingressname.dns_sub_domain`. In the example above, a virtual service called `aviingress.ns1.acme.local` will be created. Two child SNI virtual services called `ns1-aviingress-aviingress1cert` will also be created. SNI child virtual services will be of the form `namespace-ingressname-certificatename`.

Shared VIP for HTTPS with Edge Termination

In this case `shared_virtualservice_namespace` is enabled in the cloud.

The following sample Ingress file creates a HTTPS ingress with edge termination with a dedicated VIP.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: aviingress
spec:
  rules:
  - host: aviingress1.ns1.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc1
          servicePort: 80
        path: /foo
  - host: aviingress2.ns1.acme.local
    http:
      paths:
      - backend:
          serviceName: avisvc2
          servicePort: 80
  tls:
    - secretName: aviingress1cert
      hosts:
      - aviingress1.ns1.acme.local
    - secretName: aviingress2cert
      hosts:
      - aviingress2.ns1.acme.local
  backend:
    serviceName: avisvc3
    servicePort: 80
```

The above Ingress will create two child SNI virtual services for the parent virtual service called `ns1-aviingress-aviingress1cert`. SNI child virtual services will be of the form `namespace-ingressname-certificatename`.

Note: When a virtual service is created on the Avi Controller, the assigned virtual service IP for ingress services is reported back to Kubernetes.

Selective Sync of Ingress Objects

As we know, in a Kubernetes cluster, the north-south virtual services are exposed via ingress objects. These ingress objects are parsed by the cloud connector. Avi creates a corresponding north-south virtual service automatically when it detects this object in the Kubernetes cluster. Ingress objects in Kubernetes are backed by an ingress controller like NGINX Ingress Controller for Kubernetes or HAProxy Ingress Controller for Kubernetes.

Sometimes, there can be a need to selectively create objects in Avi based on the type of the controller. For example, syncing only objects that are non-NGINX based ingress objects in Avi Vantage.

Annotations in the ingress object (in Kubernetes) along with a key-value pair in Avi cloud configuration can be used to sync only a specific type of ingress objects. For example,

```
apiVersion: v1
items:
- apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    annotations:
      kubernetes.io/ingress.class: avi
    name: myapp
  spec:
    rules:
    - host: myapp.ingressdemo.avi.local
      http:
        paths:
        - backend:
            serviceName: myapp-service
            servicePort: 80
          path: /
```

In the example, `kubernetes.io/ingress.class: avi` indicates that an ingress class of type avi has been provided for Avi to sync this ingress object.

Creating an Include List

In order to set a default list, the Avi cloud object is populated with `ing_include_attributes`.

```
configure cloud `cloud name`
cloud> oshiftk8s_configuration
cloud:oshiftk8s_configuration> ing_include_attributes
cloud:oshiftk8s_configuration:ing_include_attributes> attribute kubernetes.io/ingress.class
```

```
cloud:oshiftk8s_configuration:ing_include_attributes> value avi
cloud:oshiftk8s_configuration:ing_include_attributes> save
cloud:oshiftk8s_configuration> savecloud> save
```

The value can be changed to modify the cloud object, and the corresponding change should be made in the ingress object for Avi to sync and create the north-south virtual service.

Creating an Exclude List

The knob, `ing_exclude_attributes` uses the key-value pair to create a list of objects that would be excluded for syncing.

For example, when you need to sync everything excluding the ones specified as key-value in the `ing_exclude_attributes`.

```
configure cloud `cloud name`
cloud> oshiftk8s_configuration
cloud:oshiftk8s_configuration> ing_exclude_attributes
cloud:oshiftk8s_configuration:ing_exclude_attributes> attribute kubernetes.io/ingress.class
cloud:oshiftk8s_configuration:ing_exclude_attributes> value avi
cloud:oshiftk8s_configuration:ing_exclude_attributes> save
cloud:oshiftk8s_configuration> save
cloud> save
```

Note: Both the include and exclude list can have multiple `attribute: value` pairs. In this case, Avi Vantage will create a virtual service for the ingress if at least one of the `attribute: value` pair in the list matches that on the ingress annotation.

Retaining the Current Behavior

To revert to the current behavior, modify `ing_include_attribute` to an empty value. Therefore, there is no labelling in Kubernetes object required. This will wildcard on all the ingress objects and sync everything in Avi.

Automatic Configuration

The following fields in the virtual services and pool objects are automatically derived from Kubernetes ingress by the cloud connector.

VirtualService Field	Kubernetes Field	Comments
tenant	Namespace	
name	Derived from ingress name and optionally certificate name for HTTPS	
fqdn	Host	
ip_address	Dedicated VIP: Can be explicitly specified via annotations or will be auto-allocated from the north-south IPAM object Shared VIP: Uses parent VritualServices VIP.	
services	Port field in the corresponding service	
application_profile	Layer4 if overridden via annotations or container ports don't match http_container_ports in Cloud Configuration, else Layer7	
network_profile	Default to System-TCP-Proxy unless overridden by annotations	
pool_group	One for every Service in Ingress	
pool	One for every Service in Ingress	
network_security_policy	A default network security policy is created for every new or child SNI virtual service	
http_policies	Layer7 policy for URL (host, path, port) switching amongst routes sharing the virtual service	
microservice	A microservice is automatically created for every Service	
east_west_placement	Always north-south	

Annotations

Additional configuration for Avi virtual services can be provided via annotations.

Automatic WAF policy

The following annotation automatically creates and attaches a Web application firewall (WAF) policy to the virtual service. After initial creation, the WAF policy can be modified independently of the virtual service. Note that the version field is necessary, since a WAF profile was introduced in the virtual service starting this release.

```
metadata:
  annotations:
    avi_proxy: '{"waf_policy": true, "version": "17.2.4"}'
```

If the cloud uses the shared virtual service model and just a few ingresses require WAF service, dedicated virtual services can be created just for these ingresses using the `dedicated_route` flag. The annotation would be as follows in this case.

```
metadata:
  annotations:
    avi_proxy: '{"waf_policy": true, "dedicated_route": true, "version": "17.2.4"}'
```