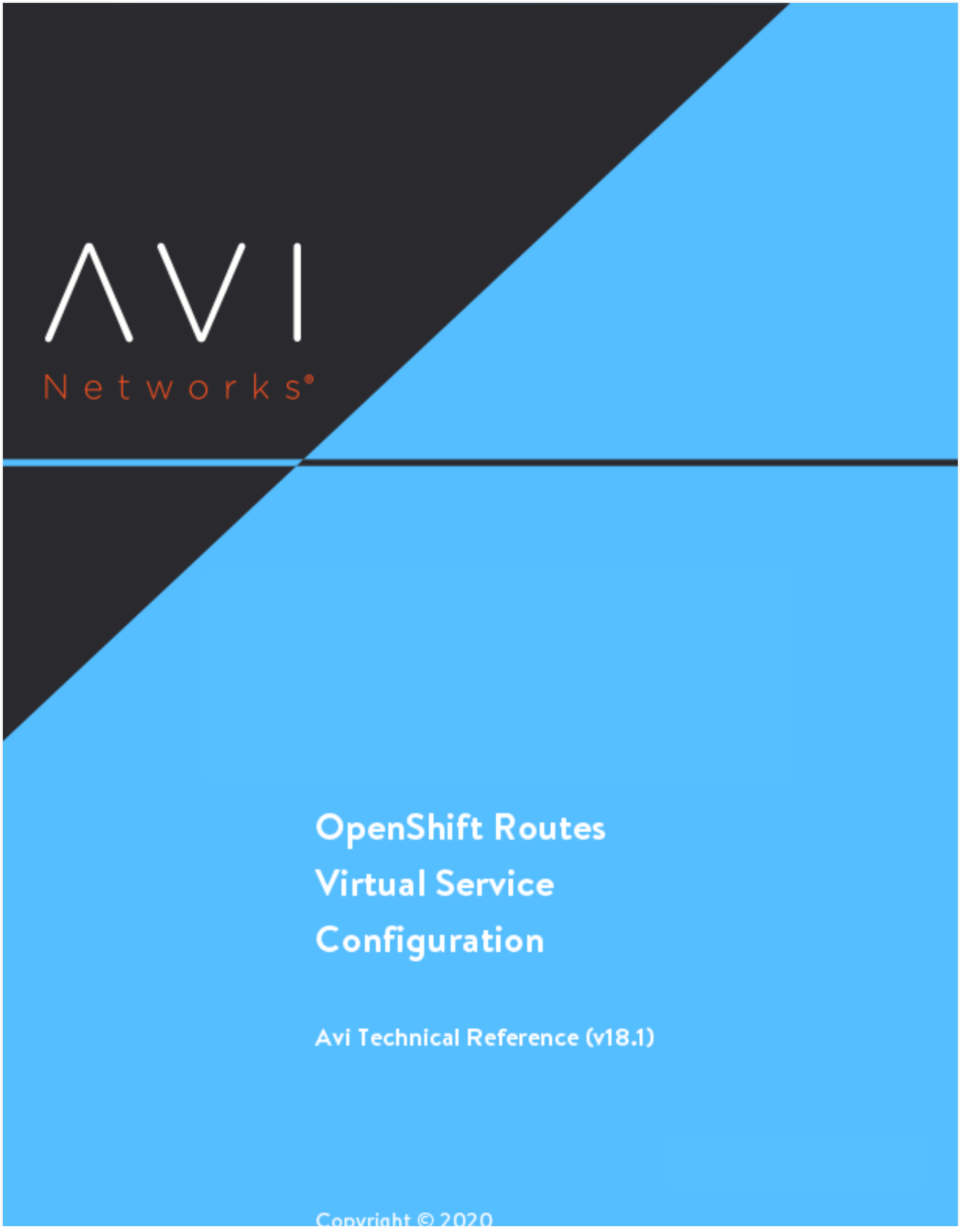
AVI
Networks®
OpenShift Routes
Virtual Service
Configuration
Avi Technical Reference (v18.1)
Copyright © 2020

OpenShift Routes Virtual Service Configuration

view online

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

OpenShift Routes to Avi Vantage Object Mapping

OpenShift routes can be configured in one of two ways as far as virtual IPs(VIP)are concerned.

- Dedicated virtual IP per route: Each OpenShift route 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 routes: Each OpenShift route 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 services 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`.

Starting with release 18.1.3, the cloud configuration `override_service_ports` has been introduced. By default, this knob is disabled. When the knob is not set, virtual service for route will use ports from OpenShift service. When the knob is set, virtual services for Route will use well known ports i.e. 80 for HTTP and 443 for HTTPS.

NB: Support for shared virtual services is available beginning 17.1.9 in the 17.1 release series and 17.2.3 in the 17.2 release series.

The following table maps each type of OpenShift route to Avi Vantage objects.

OpenShift route	Avi Vantage Object	Comment
Dedicated virtual service per route	Virtual service, Pool	Every route creates a virtual service, pool object. Route annotation *avi_proxy: '{"dedicated_route": true}'* creates a dedicated virtual service (that uses a VIP) per route in shared virtual service Projects/Namespaces.
Shared virtual service: Unsecured host/path-based routes	Pool	HTTP policy is used in parent virtual service to switch host/path to the appropriate pool.
Shared virtual service: Secure routes	Child virtual hosted/SNI virtual service, pool	SNI is used to route traffic to the appropriate child virtual service/Pool.

Unsupported Features

The following features are unsupported.

1. Passthrough termination with shared VIPs.
    1. Solution: Use dedicated VIPs instead.
2. RoutePort in RouteSpec

1. Solution: Expose a single port in deployment config.

3. East-West Load Balancing with Avi Vantage in Kubernetes/OpenShift is not supported

Shared Virtual Service Mode Configuration

- The Field `shared_virtualservice_namespace` configures the cloud to operate in either dedicated or shared VS mode. Enable Use Shared Virtual Service in the UI, if required. Enabling `shared_virtualservice_namespace` uses the shared parent virtual service for all routes in all namespaces by default. Disabling `shared_virtualservice_namespace` creates a dedicated virtual service per route in all namespaces by default.
- Changing this mode is disruptive and can be time consuming with a large number of routes; hence, it is recommended that this is 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 routes 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 routes in the namespace.
- Every route can be overridden to create a dedicated virtual service for itself, even if the cloud or the namespace is operating in the shared mode. Route annotation `avi_proxy: '{"dedicated_route": true}'` creates a dedicated virtual service for that route.

Route Configuration Examples

The following examples assume the following cloud configuration.

- `http_container_ports` is configured with 80, 8080.
- `use_service_cluster_ip_as_ew_vip` is enabled (kube-proxy is disabled cluster wide).
- `default_service_as_east_west_service` is enabled (default: on).

- 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 a 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 JSON format.

```
{
  "kind": "DeploymentConfig",
  "apiVersion": "v1",
  "metadata": {
    "name": "avitest"
  },
  "spec": {
    "template": {
      "metadata": {
        "labels": {
          "name": "avitest"
        }
      },
      "spec": {
        "containers": [
          {
            "name": "avitest",
            "image": "avinetworks/server-os",
            "ports": [
              {
                "name": "http",
                "containerPort": 8080,
                "protocol": "TCP"
              }
            ]
          }
        ]
      }
    },
    "replicas": 2,
    "selector": {
      "name": "avitest"
    }
  }
}
```

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

```
{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "avisvc",
    "labels": {
        "svc": "avisvc"
    }
  },
  "spec": {
    "ports": [
    {
      "name": "http",
      "port": 80,
      "targetPort": "http"
    }
    ],
    "selector": {
      "name": "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 Routes

Dedicated VIP for Unsecured Route

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

The following sample route file creates a unsecured route with a dedicated VIP for service avisvc.

```
apiVersion: v1
kind: Route
spec:
  host: aviroute.local
  to:
    kind: Service
    name: avisvc
metadata:
  name: aviroute
```

This dedicated virtual service route will create its own virtual service and pool with its dedicated VIP.

Shared VIP for Unsecured Route

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

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

```
apiVersion: v1
kind: Route
spec:
  host: aviroute.local
  to:
    kind: Service
    name: avisvc
metadata:
  name: aviroute
```

The unsecured application will be created as a pool for the parent virtual service. Pool name will be of the form `namespace-routename-aviroute-pool-port-protocol`. For example, route name `aviroute1` in namespace `default` will have a pool name `default-aviroute1-aviroute-pool-http-tcp`. An HTTP policy will provide host/path switching for the pool.

Dedicated VIP for Secured Route with Edge Termination With Certificate Data

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
    openshift.io/description: ""
    openshift.io/display-name: ""
    openshift.io/node-selector: region=infra
    openshift.io/requester: system:admin
    openshift.io/sa.scc.mcs: s0:c8,c7
    openshift.io/sa.scc.supplemental-groups: 1000070000/10000
    openshift.io/sa.scc.uid-range: 1000070000/10000
  creationTimestamp: 2017-08-24T23:25:13Z
  labels:
    project1: foo
  name: project1
```

```
  resourceVersion: "3845586"
  selfLink: /api/v1/namespacesproject1
  uid: 7b02ef57-8923-11e7-8756-005056b03f90
spec:
  finalizers:
  - openshift.io/origin
  - kubernetes
status:
  phase: Active
```

The following sample route file creates a secured route with edge termination with a dedicated VIP for service `avisvc`.

```
apiVersion: v1
kind: Route
spec:
  host: aviroute.local
  to:
    kind: Service
    name: avisvc
  tls:
    termination: edge
    certificate: |-
      -----BEGIN CERTIFICATE-----
      MIIDsTCCApmgAwIBAgIJANX0VbJbXVUSMA0GCSqGSIb3DQEBCwUAMG8xCzAJBgNV
     AYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQHDAtTYW50
...
      T5vbDSdlZHM1m+xOjk3jz7nQjORxtsX4qsM0He5fNu1j/76dR4qmjAgx2qPWOGfS
      1I+9RjkgA7bOKIloz28GmCAOJDHdv2ecd0UbFsXu26s0WZFE4g==
      -----END CERTIFICATE-----
    key: |-
      -----BEGIN PRIVATE KEY-----
      MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDb6YAxPwee15eI
      KwUauaQcx23yBk1wV0DhN280WHQ86OIwyPmX7MoVicOwzq0fEzw/Zu+1YfV75+GE
...
      1katRly4cgYeryEX+enH4+N0GrZepL4fHTorflL0nuZMG9HA1MLwYCK3UADqLDJ6
      45f3rxRceTmBoh+nQI5QIWk=
      -----END PRIVATE KEY-----
metadata:
  name: aviroute
```

Dedicated VIP for Unsecured Route

In this case `shared_virtualservice_namespace` is enabled in the cloud and namespace does not have the < `avi_virtualservice: dedicated` annotation.

All routes in this namespace create shared virtual services, but the following sample route file creates a unsecured route with a dedicated VIP for service `avisvc`, because the `avi_proxy: '{"dedicated_route": true}'` annotation requests a dedicated virtual service.

```
apiVersion: v1
kind: Route
spec:
  host: aviroute.local
  to:
    kind: Service
    name: avisvc
metadata:
  name: aviroute
  annotations:
    avi_proxy: '{"dedicated_route": true}'
```

This dedicated virtual service route will create its own virtual service and pool with its dedicated VIP.

Shared VIP for Route With Edge Termination With Certificate Data

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

The following sample route file creates a secured route with edge termination with VIP assigned for parent virtual service `parent-vs-Default-Group-Default-Cloud` for service `avisvc`.

```
apiVersion: v1
kind: Route
spec:
  host: aviroute.local
  to:
    kind: Service
    name: avisvc
  tls:
    termination: edge
    certificate: |-
      -----BEGIN CERTIFICATE-----
      MIIDsTCCApmgAwIBAgIJANX0VbJbXVUSMA0GCSqGSIb3DQEBCwUAMG8xCzAJBgNV
     AYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQHDAtTYW50
...
      T5vbDSdlZHM1m+xOjk3jz7nQjORxtsX4qsM0He5fNu1j/76dR4qmjAgx2qPWOGfS
      1I+9RjkgA7bOKIloz28GmCAOJDHdv2ecd0UbFsXu26s0WZFE4g==
      -----END CERTIFICATE-----
    key: |-
      -----BEGIN PRIVATE KEY-----
      MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDb6YAxPwee15eI
      KwUauaQcx23yBk1wV0DhN280WHQ86OIwyPmX7MoVicOwzq0fEzw/Zu+1YfV75+GE
...
      1katRly4cgYeryEX+enH4+N0GrZepL4fHTorflL0nuZMG9HA1MLwYCK3UADqLDJ6
      45f3rxRceTmBoh+nQI5QIWk=
      -----END PRIVATE KEY-----
metadata:
  name: aviroute
```

Secure route with a shared VIP will create a SNI-based child virtual service/pool for parent virtual service `parent-vs-Default-Group-Default-Cloud` for service `avisvc`.

Dedicated VIP Secured Route With Edge Termination, insecureEdgeTerminationPolicy Allowing HTTP Traffic With a Certificate Name

The following sample route file creates a secured route with edge termination with a reference to a certificate name instead of providing certificate data in the route file.

```
apiVersion: v1
kind: Route
spec:
  host: aviroute.local
  to:
    kind: Service
    name: avisvc
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Allow
metadata:
  name: aviroute
  annotations:
    avi_proxy: '{"virtualservice": {"ssl_key_and_certificate_refs": ["cert1"]}}'
```

Path-based Routes (dedicated mode)

In dedicated mode, a single virtual service is created for every host name. If there are multiple routes using the same host name, each route with the same host name corresponds to a pool group(s) and pool(s) for the virtual service. HTTP request policies are automatically configured to match and perform layer 7 URL switching to the appropriate pool group.

Suppose there are 2 routes ? `aviroute1` and `aviroute2` ? with host name `app.os.acme.com` and paths `/foo` and `/bar`, respectively. This will result in the following configuration in Avi Vantage.

```
VirtualService (app.os.acme.com)
  PoolGroup aviroute1-poolgroup-8080-tcp
    Pool aviroute1-pool-8080-tcp
  PoolGroup aviroute2-poolgroup-8080-tcp
    Pool aviroute2-pool-8080-tcp
```

The following sample route file creates an unsecured route for host `app.os.acme.com` and path `/foo`. This maps to a virtual service called `app.os.acme.com` and its pool group `aviroute1-poolgroup-8080-tcp`.

```
apiVersion: v1
kind: Route
spec:
  host: app.os.acme.com
  path: "/foo"
  to:
```

```
    kind: Service
    name: avisvc1
metadata:
  name: aviroute1
```

The following sample route file creates an unsecured route for host `app.os.acme.com` and path `/bar`. This maps to a virtual service called `app.os.acme.com` and its pool group `aviroute2-poolgroup-8080-tcp`.

```
apiVersion: v1
kind: Route
spec:
  host: app.os.acme.com
  path: "/bar"
  to:
    kind: Service
    name: avisvc2
metadata:
  name: aviroute2
```

Notes:
Starting with release 18.1.3,
* On assigning a virtual service on the Avi Controller, the IP of the assigned virtual service is reported back to OpenShift.
* If route sync fails, the failure status is reported.

Automatic Configuration

The following fields in the virtual services and pools objects are automatically derived from OpenShift routes by the cloud connector.

VirtualService Field	OpenShift/Kubernetes Field	Comments
tenant	Project/namespace	
name	Host name	
fqdn	Host	
ip_address	Dedicated VIP: Can be explicitly specified via annotations or auto-allocated from the north-south IPAM object Shared VIP: If parent virtual service is specified, ip_address isn?t required.	
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	Always System-TCP-Proxy	
pool_group	One for every route with the same host name	
pool	One for every route with the same host name	
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 new or child SNI virtual service
east_west_placement	Always north-south

Annotations

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

Automatic WAF policy

Starting release 17.2.4, the following annotation automatically creates and attaches a Web application firewall (WAF) policy to the VirtualService. 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 Routes require WAF service, dedicated virtual services can be created just for these Routes 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"}'
```