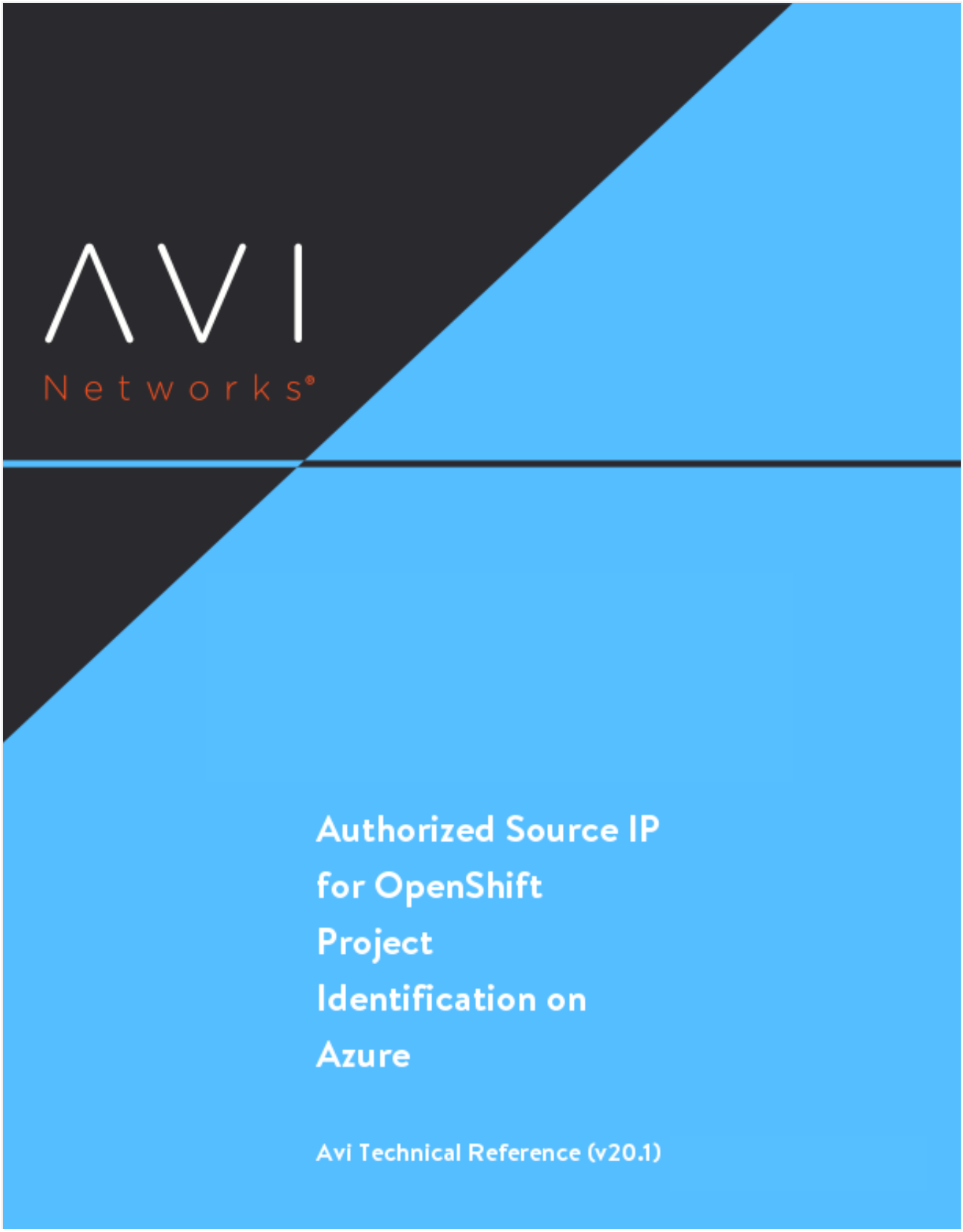


Authorized Source IP for OpenShift Project Identification on Azure

Avi Technical Reference (v20.1)

Authorized Source IP for OpenShift Project Identification on Azure

view online

Overview

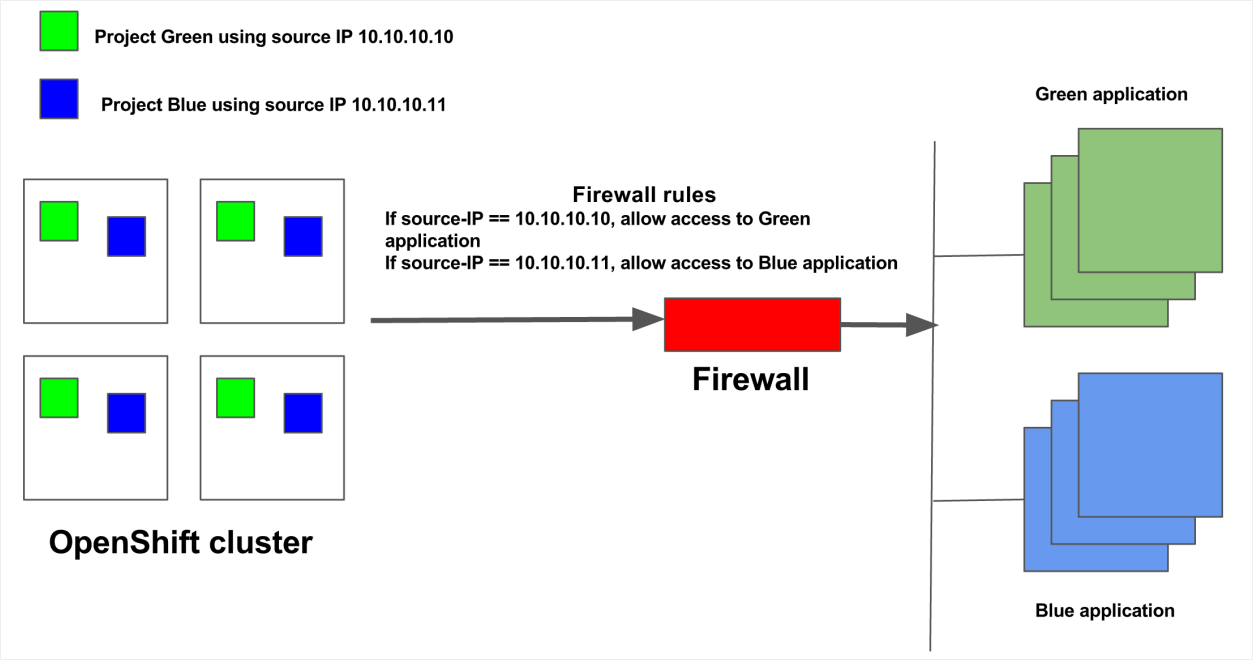
In Azure deployments, Avi Vantage can securely identify OpenShift projects using source IP address for traffic initiated from within the OpenShift cluster to external applications.

Authorized Source IP

Some deployments require identifying traffic based on the source IP address to provide differential treatment for certain applications. For instance, DMZ deployments include firewall, security, visibility, and other parameters that may require client validation before allowing traffic to any application. Such deployments can use source IP to validate the client.

The traffic initiated from within the OpenShift clusters to outside applications is masked. The actual source of the traffic is hidden from the remote application.

Consider the following example where the source IP of 10.10.10.10 securely identifies Project Green and 10.10.10.11 securely identifies Project Blue.



The network security policies in Avi Vantage prevent pods belonging to projects other than Green from using the 10.10.10.10 source IP address. Therefore, the remote application or firewall can securely identify Project Green using 10.10.10.10 source IP address.

Configuring an Authorized Source IP Instance

Prerequisites

- Avi Vantage should provide east-west services for the cluster. Refer to Configure Networks section within the OpenShift installation guide for more configuration information. > Note: There is an option to *not* use Avi for east-west services and instead use kube-proxy, with Avi handling egress-pod creation. In that case, Avi will synchronize the egress service created in K8S/OpenShift and create the egress pod for this service. Any access to the cluster IP for the egress service will hit the egress pod created by Avi without being proxied by the Avi SE. In short, Avi will manage the pods for the egress services and kube-proxy will provide load balancing to the egress pod via its egress service?s cluster IP. However, any isolation of namespaces will have to be done by a separate NetworkPolicy object in K8S.
- `secure_egress_mode` in the Avi OpenShift cloud configuration must be enabled as shown below:

```
[admin:controller]: cloud> oshiftk8s_configuration
[admin:controller]: cloud:oshiftk8s_configuration> secure_egress_mode
Overwriting the previously entered value for secure_egress_mode
[admin:controller]: cloud:oshiftk8s_configuration> save
[admin:controller]: cloud> save
```

- Authentication credentials ? OpenShift cluster access must have cluster-admin privileges (To be able to create SecurityContextConstraints and ServiceAccounts in all projects). Certificates or user-account tokens with such privileges are required to enable this feature.
- Avi Vantage needs credentials with cluster role and privileges as shown below:

```
apiVersion: v1 kind: ClusterRole metadata: creationTimestamp: 2017-04-19T22:55:04Z name: avirole resourceVersio
```

```
<li>apiGroups:</li> <li>"" attributeRestrictions: null resources:</li> <li>pods</li>
<li>projectrequests</li> <li>projects</li> <li>replicationcontrollers</li>
<li>securitycontextconstraints</li> <li>serviceaccounts</li> <li>services verbs:<
/li> <li>'*'</li> <li>apiGroups:</li> <li>"" attributeRestrictions: null resources:<
/li> <li>'*' verbs:</li> <li>get</li> <li>list</li> <li>watch</li> <li>apiGroups:<
/li> <li>"" attributeRestrictions: null resources:</li> <li>routes/status verbs:<
/li> <li>patch</li>
```

- `update`
- Identify dedicated nodes for hosting egress service pods. The nodes can be differentiated using OpenShift labels. The syntax for configuring node label is as follows:

```
 oc label node node_name key=val
```

An example for 10.1.1.1 node configured with a key value of *kb* is as shown below:

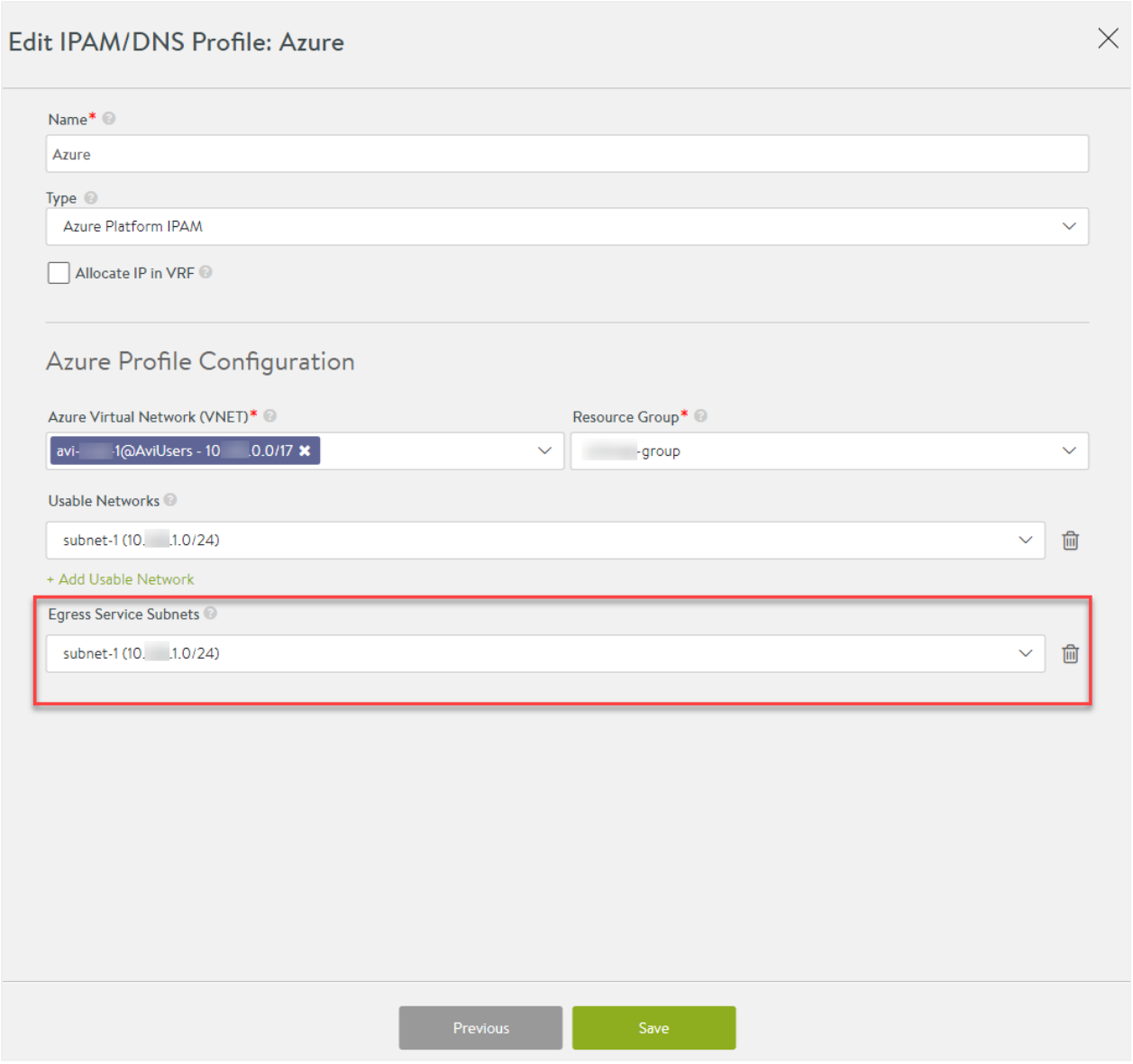
```
oc label node 10.1.1.1 k1=kb
```

- Configure the subnet of the dedicated node as the Azure egress subnet for egress source IP allocation. Based on this configuration:

- From the egress subnet, Avi Controller allocates two egress service IP addresses in the case of HA, and one egress service IP address in the case of non-HA.
- Two separate egress pods with different source IPs will spun up.
- The external service will receive requests from two different source IPs, based on which the pod serves the request.

To configure the egress service subnet on Avi UI, navigate to Templates > Profiles > IPAM/DNS Profiles and click on the Azure specific profile.

In the second tab, enter the subnet of the dedicated node under the Egress Service Subnets field.



Currently, this is restricted to just one subnet per cloud.

Workflow

You need to create one egress service per authorized source IP. To authorize multiple source IPs, it is required to create the same number of egress services within OpenShift. Avi Vantage will create a ServiceAccount for every project in OpenShift and add it to SecurityContextConstraint to enable pods to be created in privileged mode. The following code samples depict the order of configuration when a new project is created in OpenShift.

```
# oc describe scc avivantage-scc-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Name:                                   avivantage-scc-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Priority:                               [NONE]
Access:
  Users:                                system:serviceaccount:default:avivantage-bfff9603-1ffd-4bcc-aef6-118268
  Groups:                               [NONE]
Settings:
  Allow Privileged:                     true
  Default Add Capabilities:             [NONE]
  Required Drop Capabilities:           [NONE]
  Allowed Capabilities:                 [NONE]
  Allowed Volume Types:                 *
  Allow Host Network:                   true
  Allow Host Ports:                     true
  Allow Host PID:                       false
  Allow Host IPC:                       false
  Read Only Root Filesystem:            false
  Run As User Strategy: RunAsAny
    UID:                                [NONE]
    UID Range Min:                      [NONE]
    UID Range Max:                      [NONE]
  SELinux Context Strategy: RunAsAny
    User:                               [NONE]
    Role:                               [NONE]
    Type:                               [NONE]
    Level:                              [NONE]
  FSGroup Strategy: RunAsAny
    Ranges:                             [NONE]
  Supplemental Groups Strategy: RunAsAny
    Ranges:                             [NONE]
```

```
# oc describe serviceaccount avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Name:           avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Namespace:      default
Labels:         [NONE]

Image pull secrets:     avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-dockercfg-2j07a

Mountable secrets:      avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-token-7huln
                        avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-dockercfg-2j07a
```

```
Tokens:                avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-token-7huln
                       avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-token-zxi5t
```

```
# oc describe scc avivantage-scc-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Name:                                       avivantage-scc-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Priority:                                   [NONE]
Access:
  Users:                                    system:serviceaccount:default:avivantage-bfff9603-1ffd-4bcc-aef6-118268
  Groups:                                   [NONE]
Settings:
  Allow Privileged:                         true
  Default Add Capabilities:                 [NONE]
  Required Drop Capabilities:               [NONE]
  Allowed Capabilities:                     [NONE]
  Allowed Volume Types:                     *
  Allow Host Network:                       true
  Allow Host Ports:                         true
  Allow Host PID:                           false
  Allow Host IPC:                           false
  Read Only Root Filesystem:                false
  Run As User Strategy: RunAsAny
    UID:                                    [NONE]
    UID Range Min:                          [NONE]
    UID Range Max:                          [NONE]
  SELinux Context Strategy: RunAsAny
    User:                                   [NONE]
    Role:                                   [NONE]
    Type:                                   [NONE]
    Level:                                  [NONE]
  FSGroup Strategy: RunAsAny
    Ranges:                                 [NONE]
  Supplemental Groups Strategy: RunAsAny
    Ranges:                                 [NONE]
```

Configuring the egress pod involves creation of a secure service with the necessary parameters provided in the annotations. Avi Vantage uses these annotations (in the order specified below) for the following three purposes.

1. Allocating an IP address from the host network in the OpenShift cluster, as determined by the north-south IPAM configured in Avi Vantage OpenShift cloud. This IP is used as the EGRESS_SOURCE IP for the egress pod.
2. Creating egress ReplicationController with exactly one replica and the right parameters, as picked up from annotations below.
3. Updating the service selector for the secure service, so that it points to the newly created egress pod from step #2 above.

Creating a Secure Egress Service

Service Definition

The following code snippet displays service definition for a secure east-west service using the `secure-egress-service.json` command.

> *Note: In the configuration below, the use of `networksecuritypolicy` is optional. For Avi egress solution with kube-proxy, `avi_proxy` label is not required.*

```
{
 "kind": "Service",
 "apiVersion": "v1",
 "metadata": {
   "name": "secure-egress-service",
   "labels": {
       "svc": "secure-egress-service"
   },
   "annotations": {
       "avi_proxy": "{\"networksecuritypolicy\": {\"rules\": [{\"index\": 1000, \"enable\": true, \"name\": \"allowtena
       "egress_pod": "{\"destination_ip\": \"10.145.1.15\", \"nodeSelector\": {\"egress_access\": \"true\"}}"
   }
 },
 "spec": {
   "ports": [
   {
     "name": "foo",
     "port": 80
   }
   ],
   "type": "LoadBalancer"
 }
}
```

`egress_pod` is the annotation used to create the corresponding egress pod.
`destination_ip` is the destination IP address of the application outside the cluster.
Avi Vantage automatically creates a pod named `secure-egress-service-avi-egress-pod`, where `avi-egress-pod` is the suffix of the secure service name.

> *Note: `selector` is deliberately omitted from the secure service definition above, as Avi Vantage will update the secure service?s configuration once the egress pod is created successfully.*

Node Selection

Egress pods need to be restricted to dedicated nodes that have access to the north-south external network and are configured using label. This can be achieved by specifying the `nodeSelector` attribute and by adding it to egress pod annotation as `egress_pod': ``"{\"nodeSelector\": {\"external-accessible-node\": \"true\"}, \" destination_ip\": \"10.10.10.200\"}"` as specified in Kubernetes - Assigning Pods to Nodes document. In Azure, it is mandatory to use the node selector as a discriminator for different nodes.

Avi Vantage automatically creates and maintains a microservice group per project that reflects all the current pods in that project. In the above policy, the first rule allows the microservice group `default-avi-microservicegroup`, which allows all pods in the `default` project. The second rule denies all other pods from accessing the service. Thus, effectively only pods in the `default` project are allowed to access this service.

Customization

- Health monitor port:

```
"egress_pod": `"{\"hm_port\": \"1000\", \"destination_ip\": \"10.10.10.200\"}"
```

- Docker image:

```
"egress_pod": `"{\"image\": \"private-repo:5000/avi-egress-router\", \"destination_ip\": \"10.10.10.200\"}"
```

Starting with Avi Vantage release 18.2.3, customization using source IP and sources for high availability mode is supported. The newly added annotations statically assign an IP address to Egress Pod in OpenShift running in Azure.

- Source IP:

```
"annotations":
  {"egress_pod":
   "{\"destination_ip\": \"10.10.10.200\", \"source_ip\": \"10.91.107 .209\"}"}
```

- Customization for HA mode using multiple source IP addresses

```
"annotations":
  {"egress_pod":
    "{\"destination_ip\": \"10.10.10.200\", \"ha\": \"true\", \"sources\": [\"10.91.107.242\", \"10.91.107.210\"]}"}
```

Service Using OpenShift Client

Use the following command to create a service using the the OpenShift client:

```
>oc create -f secure-egress-service.json
```

Post-Secure Service Creation

Creating a secure service will trigger the following three actions on Avi Controller:

Action 1

An egress ReplicationController with the name `service-name-avi-egress-pod` is created with the below configuration. In this case, the name used is `secure-egress-service-avi-egress-pod`.

Note: The comments inserted in the code below appear as [NOTE: Comment text]

```
apiVersion: v1
kind: ReplicationController
```

```
metadata:
  creationTimestamp: 2017-05-05T20:06:40Z
  generation: 1
  labels:
    name: secure-egress-service-avi-egress-pod
  name: secure-egress-service-avi-egress-pod
  namespace: default
  resourceVersion: "2058613"
  selfLink: /api/v1/namespaces/default/replicationcontrollers/secure-egress-service-avi-egress-pod
  uid: 5a38d5dc-31ce-11e7-887a-005056b0c674
spec:
  replicas: 1
  selector:
    name: secure-egress-service-avi-egress-pod
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: secure-egress-service-avi-egress-pod
      name: secure-egress-service-avi-egress-pod
    spec:
      containers:
      - env:
        - name: EGRESS_SOURCE
          value: 10.70.112.155                [NOTE: Source IP allocated by Avi Vantage IPAM]
        - name: EGRESS_DESTINATION
          value: 10.10.24.85                  [NOTE: "destination_ip" from annotation]
        - name: BRIDGE_IP_ADDR
          value: 172.18.0.1
        - name: BRIDGE_NETMASK
          value: "16"
        - name: TCP_HM_PORT
          value: "4"                          [NOTE: "hm_port" from annotation]
        image: avinetworks/avi-egress-router   [NOTE: "image" from annotation, defaults to "avinetworks/avi-egress-rout
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          initialDelaySeconds: 10
          periodSeconds: 3
          successThreshold: 1
          tcpSocket:
            port: 4
          timeoutSeconds: 1
        name: avi-egress-router
        resources: {}
        securityContext:
          privileged: true
        terminationMessagePath: /dev/termination-log
        volumeMounts:
        - mountPath: /hostroot/var/run
          name: run
```

```
        - mountPath: /hostroot/proc/1/ns/net
          name: ns1
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      securityContext: {}
      serviceAccount: avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
      serviceAccountName: avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5    [NOTE: serviceaccount created by Avi for e
      terminationGracePeriodSeconds: 30
      volumes:
      - hostPath:
          path: /var/run
        name: run
      - hostPath:
          path: /proc/1/ns/net
        name: ns1
status:
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
```

Pod from the ReplicationController:

```
# oc describe pod secure-egress-service-avi-egress-pod-s1uhm
Name:                   secure-egress-service-avi-egress-pod-s1uhm
Namespace:              default
Security Policy:        avivantage-scc-bfff9603-1ffd-4bcc-aef6-118268e5f2b5
Node:                   10.70.112.61/10.70.112.61
Start Time:             Fri, 05 May 2017 13:06:40 -0700
Labels:                 name=secure-egress-service-avi-egress-pod
Status:                 Running
IP:                     10.129.0.229
Controllers:            ReplicationController/secure-egress-service-avi-egress-pod
Containers:
  avi-egress-router:
    Container ID:       docker://0bce263bcb1f7c0afacca23c999f0b154d416bd3c9fdbc3d0774dd868a95be7d
    Image:              avinetworks/avi-egress-router
    Image ID:           docker-pullable://docker.io/avinetworks/avi-egress-router@sha256:57907a14f6164167ae71866116c0a1
    Ports:
    State:              Running
      Started:          Fri, 05 May 2017 13:06:42 -0700
    Ready:              True
    Restart Count:      0
    Liveness:           tcp-socket :4 delay=10s timeout=1s period=3s #success=1 #failure=3
    Volume Mounts:
      /hostroot/proc/1/ns/net from ns1 (rw)
      /hostroot/var/run from run (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-token-7huln (r
    Environment Variables:
      EGRESS_SOURCE:            10.70.112.155
```

```
      EGRESS_DESTINATION:       10.10.24.85
      BRIDGE_IP_ADDR:           172.18.0.1
      BRIDGE_NETMASK:           16
      TCP_HM_PORT:              4
Conditions:
  Type          Status
  Initialized   True
  Ready         True
  PodScheduled  True
Volumes:
  run:
    Type:       HostPath (bare host directory volume)
    Path:       /var/run
  ns1:
    Type:       HostPath (bare host directory volume)
    Path:       /proc/1/ns/net
  avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-token-7huln:
    Type:       Secret (a volume populated by a Secret)
    SecretName: avivantage-bfff9603-1ffd-4bcc-aef6-118268e5f2b5-token-7huln
QoS Class:      BestEffort
Tolerations:    [NONE]
No events.
```

- EGRESS_SOURCE: Unique identifier of the project. The IP address is auto-allocated by Avi Vantage from the network in the north-south IPAM profile.
- EGRESS_DESTINATION: Destination IP address of the application outside the cluster.
- BRIDGE_IP_ADDR: IP address of the Avi bridge. The default is 172.18.0.1. This address is configured using the `avi_bridge_subnet` field in an OpenShift cloud object.
- BRIDGE_NETMASK: Netmask bits for the Avi bridge. The default is 16.
- TCP_HM_PORT: Port used for TCP health monitoring. The default is port 4, if not set. If set to a different value, change the `port` field in the `livenessProbe` section above to match this value.

The Avi egress pod has a TCP listener at port TCP_HM_PORT for health monitoring purposes. The pod is configured with a livenessProbe for health monitoring.

Action 2

The `secure-egress-service` service selector configuration is updated to reflect and use the egress pod created as shown below:

```
# oc describe service secure-egress-service
Name:                   secure-egress-service
Namespace:              default
Labels:                 svc=secure-egress-service
Selector:               name=secure-egress-service-avi-egress-pod
Type:                   LoadBalancer
IP:                     172.30.212.151
```

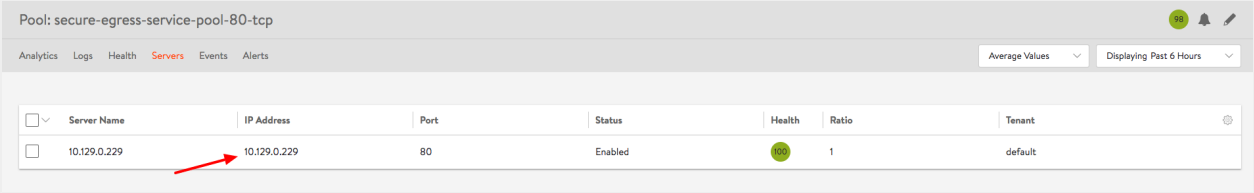
```
External IPs:           172.46.161.187
LoadBalancer Ingress:   172.46.161.187
Port:                   foo     80/TCP
NodePort:               foo     30289/TCP
Endpoints:              10.129.0.229:80
Session Affinity:       None
No events.
```

Action 3

In Avi Vantage, the secure service should be up with one pool member as the egress pod as shown below:

```
 # oc get pod secure-egress-service-avi-egress-pod-s1uhm -o wide

NAME                                        READY     STATUS    RESTARTS   AGE       IP             NODE
secure-egress-service-avi-egress-pod-s1uhm  1/1       Running   0          5h        10.129.0.229   10.70.112.61
```



Deleting the Egress Pod

The egress pod lifecycle is tied to the lifecycle of the secure service. Avi Vantage scales down the ReplicationController to zero replicas and deletes the ReplicationController for the egress pod when the secure egress service is deleted.

Deleting ServiceAccounts and SecurityContextConstraints

Service accounts created by Avi Vantage for every project will be automatically deleted when the project is deleted in OpenShift or when the OpenShift configuration is removed from Avi Vantage. SecurityContextConstraint is removed from OpenShift only when the associated cloud configuration is removed from Avi Vantage.

Service Usage

Pods in the `default` project can access the external application using the name `secure-egress-service.default.sub-domain`.

- Avi DNS will resolve `secure-egress-service.default.sub-domain` to the service virtual IP on port 80 or any other port specified in the service definition.
- Access to the virtual IP will be proxied to the secure egress Avi pod by the local Avi Service Engine.
- The secure egress Avi pod will source translate (NAT) the traffic (using the `EGRESS_SOURCE` IP address) to the remote application and use a destination IP address of `EGRESS_DESTINATION`.

The remote application will see the traffic with a source IP address of `EGRESS_SOURCE` and destination IP address of `EGRESS_DESTINATION` on port 80.

Access Patterns

The following table displays the resulting action for possible source and destination combinations:

Source	Destination	Action
Pod in default project	Service virtual IP	Allowed
Pod in default project	Secure egress Avi pod	Allowed
Pod in a different project	Service virtual IP	Denied by Avi
Pod in a different project	Secure egress Avi pod	Denied by OpenShift SDN

High Availability

When a secure Avi egress pod restarts or the host is down, OpenShift starts another instance of the pod and the service virtual IP always proxies to the right pod IP address.

You can configure high availability for the egress service through annotations, as shown in the example given: `"egress_pod": "{\"destination_ip\": \"10.145.1.15\", \"ha\": true, \"nodeSelector\": {\" egress_access\": \"true\"}}"`.