vEOS Router Configuration Guide

Arista Networks

www.arista.com

Arista vEOS version 4.20.6F
22 June 2018
Contents

Chapter 1: Overview .................................................................................................................................5

Chapter 2: vEOS Licensing .......................................................................................................................7

Chapter 3: Cloud High Availability .........................................................................................................13
  Cloud HA Topology .................................................................................................................................13
  Cloud HA Configuration ...........................................................................................................................15
  Configuring the Cloud Proxy ....................................................................................................................16
  Configuring the Cloud Provider ...............................................................................................................16
  Configuring Cloud High Availability ......................................................................................................18
  JSON-Based Cloud High Availability Configurations and Equivalent CLI Configurations ....................20
  General Troubleshooting Tips ...............................................................................................................23
  Caveats and Limitations ..........................................................................................................................23
  Cloud High Availability Commands ......................................................................................................24
  Cloud High Availability CLIs ..................................................................................................................25

Chapter 4: Using vEOS Router on the AWS Platform ............................................................................39
  vEOS Router Image Updates ....................................................................................................................39
  Amazon Machine Image (AMI) Specifications .......................................................................................39
  Supported Instance Types .......................................................................................................................39
  Methods for Launching vEOS Router Instances ....................................................................................40
    Launching vEOS Router Instances Using AWS CloudFormation .......................................................40
    Launching vEOS Router Instances Using EC2 AWS Marketplace .....................................................44
  Network Configuration Tasks for vEOS Router Instances ....................................................................53
  Using User-data for Configuration of Entities and vEOS Router Instances ...........................................57

Chapter 5: Using the vEOS Router on Microsoft Azure ........................................................................61
  vEOS Router Image Updates ....................................................................................................................61
  System Requirements .............................................................................................................................61
  Launching vEOS Router Azure Instance ................................................................................................61
    Creating an Instance using the Portal Marketplace ..............................................................................61
    Creating an Instance under Azure CLI 2.0 ..........................................................................................62
  Logging into Instance ...............................................................................................................................67
  vEOS Router Startup-Configuration using Instance Custom-Data ...........................................................68
    Sample Instance Custom-Data ..............................................................................................................69
  Troubleshooting Instance .......................................................................................................................69
  Resources ..................................................................................................................................................71

Chapter 6: Server Requirements ............................................................................................................73
  VMware ESXi Hypervisor .........................................................................................................................74
  KVM ..........................................................................................................................................................81

Chapter 7: IPsec Support ..........................................................................................................................95
  Supported Tunnel Types ..........................................................................................................................96
  Requirements when Behind a NAT ..........................................................................................................96
  Using IPsec on vEOS Router Instances ..................................................................................................97
  Topology ..................................................................................................................................................97
  Configuring IPsec Tunnels on vEOS Router Instances ............................................................................97
  Examples of Running-configurations for GRE-over-IPsec Tunnels .........................................................100
  Examples of Running-configurations for VTI IPsec Tunnels ..................................................................101
  Using IPsec on vEOS and Third Party Devices ......................................................................................103
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology</td>
<td>103</td>
</tr>
<tr>
<td>Interoperability Support</td>
<td>103</td>
</tr>
<tr>
<td>vEOS Router and Palo Alto Firewall VM</td>
<td>104</td>
</tr>
<tr>
<td>vEOS Router Show Commands</td>
<td>111</td>
</tr>
<tr>
<td>Ipsec Show Commands</td>
<td>112</td>
</tr>
<tr>
<td>vEOS Routers and CSR</td>
<td>113</td>
</tr>
<tr>
<td>CSR Configuration</td>
<td>113</td>
</tr>
<tr>
<td>Sharing IPsec Connections</td>
<td>114</td>
</tr>
<tr>
<td>IKEv1 Configuration</td>
<td>114</td>
</tr>
<tr>
<td>IKEv2 Configuration</td>
<td>115</td>
</tr>
<tr>
<td>vEOS Router (GRE-over-IPsec Tunnel)</td>
<td>117</td>
</tr>
<tr>
<td>vEOS Router (VTI IPsec Tunnel)</td>
<td>117</td>
</tr>
<tr>
<td>CSR Commands</td>
<td>118</td>
</tr>
<tr>
<td>CSR Router Show Commands</td>
<td>118</td>
</tr>
<tr>
<td>vEOS Routers and AWS Specific Cloud Configuration</td>
<td>121</td>
</tr>
<tr>
<td>IPsec Between the vEOS Router and AWS Specific Cloud Configuration</td>
<td>121</td>
</tr>
<tr>
<td>Running-configuration of the vEOS Router and AWS Specific Cloud</td>
<td>121</td>
</tr>
<tr>
<td>AWS Specific Cloud Configuration</td>
<td>122</td>
</tr>
<tr>
<td>AWS Specific Cloud Configuration Modifications</td>
<td>122</td>
</tr>
</tbody>
</table>

**Chapter 8: ECMP**

Adding ECMP: 125
Overview

vEOS Router

Arista vEOS Router is a new platform release of EOS that is supported on Amazon Web Service (AWS), Microsoft Azure and other public clouds. It is also supported on customer equipment running Linux and VMware hypervisors. By bringing advanced network telemetry and secure IPSec VPN connectivity in a software-only package, vEOS Router provides a consistent, secure and universal approach to hybrid cloud networking for any virtualized cloud deployment. Use cases for vEOS Router include Secure Multi Cloud Connectivity, Interconnecting VPCs/VNets in the Public Cloud, Multi-site VPN aggregation and Network Function Virtualization.
vEOS Licensing

Licensing for vEOS

There are two licenses available as a software subscription which must be applied to the vEOS Router software after an instance is launched for the activation of all capabilities:

• vEOS Router license - Unlocks the instance from the default performance limit of 80 Mbps.
• IPsec license

**SS-VEOSR-IPSEC-500M-1M**

The vEOS Router SW Subscription License for a single vEOS instance for 1-Month for up to 500Mbps throughput. This includes base routing features, IPsec encryption and SW support.

**SS-VEOSR-IPSEC-1G-1M**

The vEOS Router SW Subscription License for a single vEOS instance for 1-Month for up to 1Gbps throughput. This includes base routing features, IPsec encryption and SW support.

**SS-VEOSR-IPSEC-10G-1M**

The vEOS Router SW Subscription License for a single vEOS instance for 1-Month for up to 10Gbps throughput. This includes base routing features, IPsec encryption and SW support.

If a valid license has never been installed,

• The performance of the instance is limited to 10Mbps.
• IPsec is not available without a license.

For purchased licenses, upon expiration or nearing expiration,

• Renew the license as you would renew a service agreement. (The performance of the vEOS Router and IPsec instance are not impacted).
• If the license is renewed, there is no impact of service, provided there is an overlap of license dates.

Support for Bring-Your-Own-License (BYOL)

Bring your own license (BYOL) is only supported on AWS. The pricing on AWS includes both the AWS instance cost and the Arista license fee.
Installing Licenses

Licenses are files that are imported via the CLI. Contact your local SE for assistance in obtaining a license. Use the `license import` command to download a license file. Save the file to /mnt/flash/ or a server. For example purposes, the licenses below are non-functional.

veos#license import flash:vEOSLic-1.json
veos#license import flash:IPSecLic-1.json

Verifying Installed Licenses

Use the `show license` command to display details regarding the active licenses and device-specific information needed for licensing. For example purposes, the licenses below are non-functional.

veos#show license
System Serial number: 2BC6A772072B04BED43DCCF8777F036F
System MAC address: 06:1b:8a:48:8d:0c
Domain name: Unknown
License feature: IPSec
License parameter: None
Count: 1
Start: 2017-09-18 13:56:45
Expiration: 2017-12-30 16:00:00
Active: yes

License feature: vEOS - Virtualized EOS
License parameter: None
Count: 1
Start: 2017-10-08 17:00:00
Expiration: 2017-12-30 16:00:00
Active: yes

Update License (Optional)

Use the `license update` command to trigger an update of licenses in storage.

veos#license update

Obtaining and Installing Soft Expiry

Users can obtain licenses from Arista that extend the time for which the customer can use a certain feature without any limitations. The license for the feature is considered expired, but the feature continues to work until the grace period as mentioned in the license lapses.

For example, with a license such as the one below, customer can continue to use vEOS without any limitations for ten days beyond expiry date.

```json
{
    "LicenseFileVersion": "1.0",
    "CustomerName": "Arista Test Customer",
    "LicenseSerialNumber": "ARISTA-TEST-DAYSPAST1",
    "Signature": {
        "SigningCertPEM": "-----BEGIN CERTIFICATE-----7brkfssZDrRIatxKEkv6Oc
hmq6sfk2eFBUYIr2Bm9RUbVbyLZLCOv2KxJ7FF29lv1jp5An\nAyHLJUMq9qw/kvUUvUq1b1/PtE01Nc9Nd7
3yeh\nHByzTw8/f+gjKkUjQpVnucugSnkFotBPNNj/LjbQ40R/tJ0z/8sXCGJuo4mE9s/MwnWmkAhxpZyCcMBl1p3LkJk/nFHcsVb36Vc1v5WDe5AxU+0sQjEBLGp7nYo8wjjvSZIpYXRiAmDRGuAGl/W/W3F\n6hEQ661JK4KPJvoQsMqYYa0/TkZP1XEAdgEDkmj0=-----END CERTIFICATE-----",
    "Hash": ...
}
vEOS Licensing

Additional Licensing Show Commands

The following CLIs can be used to verify if a license is valid, when it expires, what licenses are installed and any relevant information regarding a license. The **show license** commands do not list features that are unlocked by external licenses or means and does not list the pay-as-you-go license provided by AWS.

Show License Files

Use the **show license files** command to display all information related to the active licenses installed. For example purposes, the licenses below are non-functional.

veos#show license files

License name: 2017.11.02.08.23.23.053684_IPSecLic-1yr.json

Contents:

```
"BindingInfo": {
   "DomainAddress": "",
   "SerialNumber": "C3F3580316A92EE8D97DB70C967EAAA4",
   "SystemMAC": "02:9c:a8:a5:51:5a"
},
"CustomerName": "Arista Test",
"Features": {
   "IPSec": {
      "Count": 1,
      "Valid": {
         "NotAfter": "2018-12-31T00:00:00Z",
         "NotBefore": "2017-11-02T15:21:22Z"
      }
   }
}
```
show license files compressed

Use the `show license files compressed` command to display license information. In this example, the files are zipped then base64 encoded. For example purposes, the licenses below are non-functional.

```bash
veos#show license files compressed
License name: 2017.11.02.08.23.23.053684_IPSecLic-1yr.json
Contents: (truncated)
```

show license expired

The `show license expired` command will display the same as the `show license` command, but with expired licenses only displayed.

```bash
veos#show license expired
System Serial number: 2BC6A772072B04BED43DCCF8777F036F
System MAC address: 06:1b:8a:48:8d:0c
Domain name: Unknown
License feature: IPSec
License parameter: None
Count: 1
Start: 2017-10-05 21:49:13
Expiration: 2017-10-09 17:00:00
Active: expired
```
show license all

The `show license all` command will display all licenses that are active, expired or licenses that have not been activated yet.

```
veos#show license all
System Serial number: 2BC6A772072B04BED43DCCF8777F036F
System MAC address: 06:1b:8a:48:8d:0c
Domain name: Unknown

License feature: IPSec
License parameter: None
Count: 1
Start: 2017-12-30 16:00:00
Expiration: 2018-12-30 16:00:00
Active: in future

License parameter: None
Count: 1
Start: 2017-09-18 13:56:45
Expiration: 2017-12-30 16:00:00
Active: yes

License parameter: None
Count: 1
Start: 2017-10-05 21:49:13
Expiration: 2017-10-09 17:00:00
Active: expired

License feature: vEOS - Virtualized EOS
License parameter: None
Count: 1
Start: 2017-10-08 17:00:00
Expiration: 2017-12-30 16:00:00
Active: yes

License parameter: None
Count: 1
Start: 2017-12-30 16:00:00
Expiration: 2018-12-30 16:00:00
Active: in future

License parameter: None
Count: 1
Start: 2017-10-05 21:49:13
Expiration: 2017-10-09 17:00:00
Active: expired
```
Chapter 3

Cloud High Availability

In the cloud, resources can be deployed across different regions or multiple locations within a region for fault tolerance reasons. AWS Availability Zones and Azure Availability Sets (or Fault Domains; Azure currently supports different resource groupings within a physical datacenter) are examples of cloud high availability offerings. When deploying vEOS Routers to enhance your cloud’s network capability, deploy the vEOS Routers as a high availability pair using the vEOS Cloud High Availability feature that fits your cloud's high availability design.

The Cloud High Availability (Cloud HA) feature adds support to make the vEOS Router deployment more resilient to various failure scenarios in the cloud, such as:

- vEOS Router instance goes down due to underlying cloud infrastructure issues.
- vEOS Router instance is unable to forward traffic due to connectivity issues in the cloud infrastructure.
- vEOS Router experiences an internal issue leading to unavailability.

vEOS Router HA pair with Cloud HA is an active-active deployment model for different cloud high availability design in a region. Each vEOS Router in an HA pair provides enhanced routing capabilities as the gateway (or next-hop router for certain destinations) for the subnets to which the vEOS routers connect. The two vEOS Router peers monitor the liveness of each other by using Bidirectional Forwarding Detection (BFD) between the router interfaces. In case of the cloud infrastructure issues or vEOS router failure, the active vEOS router takes over as the gateway or next-hop for the subnets that were connected to the peer router through cloud-specific API calls that modify the corresponding cloud route table(s) according to pre-configured information.

Cloud HA Topology

This diagram shows an example of a vEOS Router Cloud HA implementation.
In the diagram above, a virtual network is a collection of resources that are in the same cloud region. Within this virtual network, the resources, including vEOS routers, deploy into two cloud high availability zones (Availability Zones for AWS and Fault Domain for Azure) for fault tolerance reasons.

**Note:** For ease of discussion, we will use availability zone 1 and 2 to reference the high availability design in different clouds going forward.

Within each availability zone, the hosts/VMs and vEOS interfaces are connected to their corresponding subnets when the network is operating normally. Each subnet associates to a route table within the cloud infrastructure. Static routes are configured in the cloud route tables so the traffic from the hosts/VMs are routed to vEOS Routers in the corresponding availability zone as gateway or next-hop to reach certain destinations. For example, configure a default route (0.0.0.0/0) in the cloud route table with the next-hop as vEOS Router’s cloud interface ID or IP (varies depending on the cloud). The routing policy or protocol, such as BGP, on the vEOS Routers, are user configurable based on user’s network design.
The two vEOS Routers in the diagram above are configured with the Cloud HA feature as HA peers. The Cloud HA on the vEOS routers would establish a BFD peering session between the two devices through ethernet or tunnel interfaces.

When BFD connectivity loss is detected by the active vEOS router, the existing routes in the backup route table in the cloud would be updated through cloud-specific API to use the active vEOS router as the next-hop. For example, if vEOS 2 detected BFD connectivity loss with its peer, vEOS 2 would update the routes in Route Table 1 so traffic from hosts in Subnet 1 and Subnet 2 for vEOS 1 would be forwarded to next-hop ID or IP owned by vEOS 2. Traffic from the hosts in availability zone 1 would first be forwarded to the corresponding subnet gateways in the cloud. After that, the subnet gateways in the cloud would forward the traffic toward the new next-hop interface ID or IP that exist on vEOS 2. When vEOS 2 received the traffic, it would forward the traffic on according to its routing table.

What about traffic going toward the hosts in availability zone 1 while connectivity to vEOS 1 is down? When connectivity to vEOS 1 is down, hosts behind Subnet 1 and Subnet 2 become unreachable to the other part of the network (routes being withdrawn by routing protocols like BGP). Since Subnet 1 and Subnet 2 are not directly connected to vEOS 2, a routing strategy for the two subnets as "backup" on vEOS 2 is to be considered as part of your network design. A typical design would be to use static routes for the subnets connected to the peer vEOS router and point them toward the cloud subnet gateways of the active vEOS router (for example, static route for peer subnet 10.1.1.0/24 would be configured on the active vEOS router as ip route10.1.1.0/24 10.2.1.1 255 where 10.2.1.1 is the gateway/next-hop for one of the ethernet interfaces) with a high administrative distance value (least preferred). The static routes would be redistributed or advertised when the original routes with better administrative distance are withdrawn or removed by dynamic routing protocol (such as BGP).

When BFD peering session is restored to UP state upon recovery, each active vEOS router would restore its locally controlled route table entries (per user configuration) to point to itself as primary gateway again.

Cloud HA Configuration

This example configuration is based on the Cloud HA implementation diagram. The point of reference of the configuration is the vEOS Router instance vEOS 1 in the Gateway Virtual Network.

Note: Starting from Release 4.20.6, the Cloud HA configuration is only available through the CLI. The JSON file from the previous vEOS version is deprecated. You must convert the JSON configuration to CLI configuration after upgrading from any previous vEOS version. For information regarding the conversion of the JSON configuration to CLI configuration, go to JSON-Based Cloud High Availability Configurations and Equivalent CLI Configurations on page 20.

Cloud HA Modes

The Cloud HA related configurations are divided into three separate configuration modes:

- **Cloud Proxy** - For proxy related configuration such as http and https.
- **Cloud Provider** - For cloud provider specific configuration such as region, credential, and proxy name.
- **Cloud High-Availability** - For configurations such as route, next-hop, BFD source interface, and peer.

The example includes specific configurations for various aspects of the Cloud HA implementation that are configured prior to implementation. The specific configurations are:

- Configuring the Cloud Proxy on page 16
- Configuring the Cloud Provider on page 16
- Configuring Cloud High Availability on page 18
Configuring the Cloud Proxy

Optional proxies can be configured if used in a deployment. The configuration is applicable for any cloud type. All web traffic for the underlying restful APIs for the Cloud provider SDK will use the configured proxies. Multiple proxies can be configured but only one can be used at any given time from the Cloud High-Availability configuration.

```plaintext
veos(config)#
veos(config)#cloud proxy test
veos(config-cloud-proxy-test)#
```

The following example configures the cloud proxy IP, port, and username and password for HTTP.

```plaintext
veos(config)#
veos(config)#cloud proxy test
veos(config-cloud-proxy-test)#http 1.2.3.4 1234 username test password 7075E731F1A
veos(config-cloud-proxy-test)#
```

Configuring the Cloud Provider

The following describes configurations required for Cloud HA on different types of clouds.

Cloud Configuration

To have access to the cloud services, the vEOS Router must be provided with credentials. Additionally, a proxy may be configured for the connection to the cloud services to go through.

AWS Specific Cloud

Complete the following tasks to configure AWS Specific Cloud services.

- Configure Credentials
- Access to AWS Specific Cloud API Server
- If vEOS is associated with a public IP address, no special configuration is required.
- If vEOS is not associated with an public IP address, either use AWS Private Link or Proxy configuration

Configure Credentials

In the AWS Specific Cloud configuration, a region must be specified. It is recommended to authorize the vEOS Router by assigning it an IAM role, but an explicit credential can also be specified.

- IAM Role Configuration - No credentials. See Cloud Provider Helpful Tips on page 18 for additional information.
- Explicit Credential Configuration

AWS Specific Cloud IAM Role Configuration

The IAM role should be configured on the AWS Specific as shown below. This is the recommended configuration.

- "Trust Relationships" has "ec2.amazonaws.com" as trusted entities.
• "Policy" with "Permissions" for the network related EC2 actions.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ec2:AssociateRouteTable",
            "ec2:CreateRoute",
            "ec2:CreateRouteTable",
            "ec2:DeleteRoute",
            "ec2:DeleteRouteTable",
            "ec2:DescribeRouteTables",
            "ec2:DescribeVpcs",
            "ec2:ReplaceRoute",
            "ec2:DisassociateRouteTable",
            "ec2:ReplaceRouteTableAssociation",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeInstances",
            "ec2:DescribeSubnets"
         ],
         "Resource": "*"
      }
   ]
}
```

This is applicable only when running in AWS cloud environment and configures various aspects of Cloud HA feature to interact with AWS web services.

**Note:** The **access-key-id** and **secret access-key** commands are either both configured or both are omitted. If omitted, the Cloud HA Agent will try to use AWS IAM role for security tokens to access and control AWS route tables. Verify the IAM role for the vEOS router Virtual Machine (VM) is configured properly on the AWS cloud. Refer to AWS documentation to configure IAM role.

```bash
veos(config)# cloud provider aws
veos(config-cloud-aws)# access-key 0 ATPAILIL5E982IPT7P3R
veos(config-cloud-aws)# secret access-key 0 M0RRUtAA8I8wYxJB8
veos(config-cloud-aws)# region us-west-1
veos(config-cloud-aws)# proxy test
```

Configure the **backup-gateway**, **primary-gateway**, **Route Table ID**(rtb) and **local interface** for AWS.

The Route Table ID specifies for AWS the backup-gateway and primary gateway, then the destination selects the individual route within the route table to control. The **local-cloud-interface** then points to the interface ID **eni-867caa86** (from AWS perspective) of the vEOS router that the traffic should be directed.

```bash
veos(config)# cloud high-availability
veos(config-cloud-ha)# peer veos2
veos(config-cloud-ha-peer-veos2)# aws
veos(config-cloud-ha-peer-veos2-aws)# backup-gateway rtb-40b72d24 0.0.0.0/0 local-cloud-interface eni-867caa86
veos(config-cloud-ha-peer-veos2-aws)# primary-gateway rtb-2843124c 0.0.0.0/0 local-cloud-interface eni-867caa86
```

**Explicit Credential Configuration**

The explicit credential should be configured as shown below.

```bash
veos(config)# cloud provider aws
veos(config-cloud-aws)# region us-west-1
veos(config-cloud-aws)# access-key 0 MYEXAMPLESECRETKEY
```
Azure

There are two authorization models that can be used in Azure: SDK Auth Credentials and Active Directory Credentials. SDK Auth Credentials are the recommended authorization model.

• **SDK Auth Credentials**

To generate SDK Auth Credentials, use the `sdk authentication credential-file flash:startup-config` command in the `config-cloud-azure` configuration mode.

```
veos(config)#cloud provider azure
veos(config-cloud-azure)#sdk authentication credential-file flash:startup-config
```

• **Active Directory Credentials**

The following example places the vEOS router into the `config-cloud-azure` configuration mode and sets the active directory credentials.

```
veos(config)#cloud provider azure
veos(config-cloud-azure)#active-directory credential
email subscription-id ef16892c-aa46-4aba-ae9a-d4fhsblc612c
```

**Cloud Provider Helpful Tips**

The following are needed for Cloud High Availability but are not part of the vEOS configuration on the vEOS Router. These may change or can be another way to achieve the same effect without changing the vEOS Router.

**AWS VPN Specific Cloud PrivateLink**

AWS VPN Specific Cloud PrivateLink allows a private (no public IP address) vEOS instance to access services offered by AWS (without using proxy).

The interface VPC endpoints enables a private vEOS instance to connect to AWS VPN Specific Cloud PrivateLink.

To configure Interface VPC Endpoints:

1. Open the Amazon VPC console and choose **Endpoints** in the navigation panel.
2. Select **Create Endpoint**.
3. Choose the **AWS Services** and select service name `com.amazonaws.<your-region>.ec2`.
4. Choose the VPC and the subnets in each availability zone for the Interface VPC endpoints.
5. Enable private DNS name and set security group accordingly.
6. Select **Create Endpoint**.

Once the Endpoint(s) is created, the EC2 API IP associated with the domain-name will be updated to the endpoint IP.

Additional interface VPC endpoints information can be found at: [https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpce-interface.html](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpce-interface.html)

**Configuring Cloud High Availability**

To enable the Cloud HA and its parameters, use the following configurations.

**Enable Cloud High Availability**
The **cloud high-availability** command places the vEOS in the **cloud-ha** configuration mode. This example enables cloud high-availability and configures the peer **veos2**.

```plaintext
veos(config)#cloud high-availability
veos(config-cloud-ha)#no shutdown
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#
```

### Configuring BFD

To configure the BFD link between the HA pair of vEOS Routers that is used to detect peer failure, the peer IP address and local BFD source interface must be provided. The following example configures Tunnel 2 as a single hop for the source interface for BFD.

```plaintext
veos(config)#cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#bfd source-interface tunnel 2 single-hop
```

### Configuring the Recovery Time

The **recovery wait-time** command in the **cloud-ha** configuration sub-mode configures the amount of time to take back control of local route tables after failure recovery. The following example shows the wait time is configured to 90 seconds.

```plaintext
veos(config-cloud-ha-peer-veos2)#recovery wait-time 90
```

### Full Configurations

#### AWS VPN Specific Cloud Full Configuration

The following AWS configuration is valid for use with the IAM role.

```plaintext
cloud provider aws
    region us-west-1
!
cloud high-availability
    no shutdown
!
peer veos2
    aws
        backup-gateway rtb-40b72d24 0.0.0.0/0 local-cloud-interface eni-26cb1d27
        backup-gateway rtb-17b32973 0.0.0.0/0 local-cloud-interface eni-1589e714
        backup-gateway rtb-54503330 0.0.0.0/0 local-cloud-interface eni-56cf1957
        primary-gateway rtb-a4be24c0 0.0.0.0/0 local-cloud-interface eni-26cb1d27
        primary-gateway rtb-40b72d24 0.0.0.0/0 local-cloud-interface eni-56cf1957
        primary-gateway rtb-63b02a07 0.0.0.0/0 local-cloud-interface eni-1589e714
    peer address 10.2.201.149
    recovery wait-time 5
        bfd source-interface# Ethernet1
!
```

#### Azure Full Configuration
The following Azure configuration is valid for the MSI.

cloud high-availability
no shutdown
!
peer veos2
azure
backup-gateway Subnet-2-vEOS-RouteTable 0.0.0.0/0 10.1.2.4 resource-group CloudHaAzure
backup-gateway Subnet-2-vEOS-RouteTable 10.1.0.0/16 10.1.2.4 resource-group CloudHaAzure
backup-gateway Subnet-3-vEOS-RouteTable 10.1.0.0/16 10.1.3.4 resource-group CloudHaAzure
backup-gateway Subnet-3-vEOS-RouteTable 0.0.0.0/0 10.1.3.4 resource-group CloudHaAzure
primary-gateway Subnet-1-vEOS-RouteTable 10.1.0.0/16 10.1.1.4 resource-group CloudHaAzure
primary-gateway Subnet-1-vEOS-RouteTable 0.0.0.0/0 10.1.1.4 resource-group CloudHaAzure
peer address 10.1.0.5
recovery wait-time 10
bfd source-interface Ethernet1

---

**JSON-Based Cloud High Availability Configurations and Equivalent CLI Configurations**

**Note:** Starting from 4.20.6, the Cloud HA configuration is only available through the CLI. The JSON file from the previous vEOS version is deprecated. You must convert the JSON configuration to CLI configuration after upgrading from any previous vEOS version.

**Mapping JSON Config to the New CLI**

Use the following to map the previous JSON file to the new CLI.

**Mapping JSON Config to Cloud High-Availability**

The following JSON Configurations are now available in Cloud High-Availability configuration mode.

- generalConfig
- bfdConfig
- awsConfig
- azureConfig
- awsLocal/PeerRoutingConfig
- azureLocal/PeerRoutingConfig

**AWS JSON Configuration Example**

```json
"generalConfig" : {
    "enable_optional" : "true",
    "hysteresis_time_optional" : "10",
    "source_ip_optional" : "10.10.1.1"
},
"bfdConfig" : {
    "peerVeosIp" : "10.10.1.2",
    "bfdSourceInterface" : "Tunnel1"
}
```
"awsLocalRoutingConfig" : {
    "routeTableIdAndRouteNetworkInterface" : [
        { "routeTableId" : "rtb-12345678", "destination" : "0.0.0.0/0", "routeTarget" : "eni-12345678" }
    ],
},
"awsPeerRoutingConfig" : {
    "routeTableIdAndRouteNetworkInterface" : [
        { "routeTableId" : "rtb-87654321", "destination" : "0.0.0.0/0", "routeTarget" : "eni-12345678" }
    ],
}

AWS Equivalent CLI Configuration

cloud high-availability
   no shutdown
   !
   peer veos2
   aws
      backup-gateway rtb-87654321 0.0.0.0/0 local-cloud-interface eni-12345678
      primary-gateway rtb-12345678 0.0.0.0/0 local-cloud-interface eni-12345678
      peer address 10.10.1.2
      recovery wait-time 10
      bfd source-interface Tunnel1 single-hop

Azure JSON Configuration

"generalConfig" : {
    "enable_optional" : "true",
    "hysteresis_time_optional" : "10",
    "source_ip_optional" : "10.10.1.1"
},
"bfdConfig" : {
    "peerVeosIp" : "10.10.1.2",
    "bfdSourceInterface" : "Tunnel1"
},
"azureLocalRoutingConfig" : {
    "resourceGroupName" : "resourceGroup1",
    "routeTables" : [
        { "routeTableName" : "Subnet-vEOS1-RouteTable", "routes" : [ { "prefix" : "0.0.0.0/0", "nextHopIp" : "10.1.2.4" } ] }
    ],
},
"azurePeerRoutingConfig" : {
    "resourceGroupName" : "resourceGroup1",
    "routeTables" : [
        { "routeTableName" : "Subnet-vEOS2-RouteTable", "routes" : [ { "prefix" : "0.0.0.0/0", "nextHopIp" : "10.1.2.4" } ] }
    ]}
Azure Equivalent CLI Configuration

```
cloud high-availability
  no shutdown
  peer veos2
    aws
      backup-gateway Subnet-vEOS2-RouteTable 0.0.0.0/0 10.1.2.4
      resource-group resourceGroup1
      primary-gateway Subnet-vEOS1-RouteTable 0.0.0.0/0 10.1.2.4
      resource-group resourceGroup1
      peer address 10.10.1.2
      recovery wait-time 10
      bfd source-interface Tunnell single-hop
```

**Mapping JSON Config to the Cloud Provider**

The following JSON configurations are now available in Cloud Provider configuration mode.

- region
- aws_credentials_optional
- azureSdkAuthCredentials

The example below uses the AWS access key and proxy.

**AWS JSON Configuration**

```
"region" : "us-west-1",
"aws_credentials_optional": {
  "aws_access_key_id" : "ABCDEFGHIJKLMNOPQRST",
  "aws_secret_access_key" : "TSRQPONMLKJIHGFEDCBA"
```

**AWS Equivalent CLI Configuration**

```
cloud provider aws
  region us-west-1
  access-key-id 7 1234567890ABCDEFGHIJKLMNOPQRST
  secret access-key 7 1234567890TSRQPONMLKJIHGFEDCBA
  proxy proxy1
```

**Mapping JSON Config to the Cloud Proxy**

The following JSON configurations are available in Cloud Proxy configuration mode.

**Note:** In the Cloud HA CLI, the Cloud Proxy name must be referenced in the Cloud Provider Proxy configuration to use the proxy.

**JSON Configuration**

```
"http_proxy_optional": {
  "http_port_optional" : "443",
  "http_proxy_port_optional" : "8888",
  "http_proxy_optional" : "10.3.3.3",
  "http_proxy_user_optional" : ",",
  "http_proxy_password_optional" : ""
}
```

**Equivalent CLI Configuration**

```
cloud proxy proxy1
  https 10.3.3.3 8888
```
General Troubleshooting Tips

If the Cloud HA feature is not working as expected, follow these tips for debugging.

• Make sure that the network connectivity is there and DNS server is setup correctly for this feature to work.
• If using Proxy and IAM role under AWS, make sure that the HTTP traffic (TCP port 80) is not proxied to allow for temporarily security credentials to be retrieved by vEOS instance.
• Make sure to use a corresponding BFD source interface on the peer vEOS instance. This makes sure that the BFD traffic ingress and egress are on the same interface on each instance.
• For an AWS Specific Cloud, if the IAM role does not work, Arista recommends temporarily using access-key id and secret access key with enough permissions to make sure the rest of the Cloud HA configuration is fine until you debug IAM role policy.

Caveats and Limitations

• This feature was introduced in EOS release 4.20.5F which uses /mnt/flash/cloud_ha_config.json file for Cloud HA configuration without any CLI support. Starting from release 4.20.5.A1 onwards, Cloud HA feature supports CLI based configuration only. Deployments using JSON based config are not supported and will not work when the image is upgraded or downgraded. To upgrade image, the administrator must configure Cloud HA feature manually by converting the JSON config to equivalent CLI configuration. Downgrading will work as long as the older jJSON file is still present in /mnt/flash directory.
• Only a single resource-group is supported across all routing entries for Azure under Cloud specific config HA configuration.
• Cloud HA feature currently supports only a single peer.
• The AWS IAM role or Azure MSI needs to be configured properly using cloud provider's management tools and should give sufficient permissions to vEOS instance to access and update route table entries.
• The vEOS instance should have connectivity to the cloud provider's web services. The access can also be via proxy or using feature like AWS private-link.
• The recovery wait-time should not be configured less than 10 sec to avoid unnecessary route flapping when experiencing periodic instabilities.
• The Cloud HA feature will completely validate all the provided cloud configuration to make sure it is consistent and has all required permissions. However, the administrator should not change the provider's network configuration afterwards to avoid any issues during fail-over.
• When there are BFD connectivity issues between the two vEOS peers, each instance will take over the other's traffic. This cross traffic forwarding on provider's network should not have any adverse affect and still work as active-active even though both of the instance will report as Fail-over. After the network connectivity is resolved, the traffic pattern should revert to normal active-active mode.
• The user can adjust the BFD specific parameters for the session used by Cloud HA feature using normal BFD commands such as multiplier, tx/rx intervals etc. The Cloud HA fail-over and traffic takeover time is directly correlated with BFD failure detection time. However, when using an overly aggressive BFD, the failover time may incur higher overhead as well may result in greater instability during traffic bursts. Arista recommends using the use default BFD interval which is currently 300 msec with multiplier as 3.
• The bfd source-interface used in Cloud HA configuration should not belong and/or routable via the route-tables controlled by the vEOS router instance itself to avoid traffic looping issues.
• If the Cloud HA is in an invalid configuration state due to erroneous/mismatched configuration in the provider's cloud, the administrator has to force update the Cloud HA configuration (for example by shut/no shut under Cloud HA mode) after updating the provider's cloud configuration. In other words, by itself the Cloud HA feature will not retry the back-end configuration check if it is found to be invalid at the time of configuration
Cloud High Availability Commands

Global
• cloud high availability cloud high-availability (vEOS) on page 28
• cloud high availability shutdown cloud high-availability shutdown (vEOS) on page 28
• cloud high availability peer cloud high-availability peer

Interface
• backup-gateway backup-gateway (vEOS - Azure) on page 27
• bfd-source_interface bfd source-interface (vEOS) on page 27
• peer peerName peer (vEOS) on page 32
• primary-gateway (Azure Submode) primary gateway (vEOS - Azure) on page 33
• recovery-wait-time recovery wait-time (vEOS) on page 34

Cloud Provider Commands

Global
• cloud provider azure cloud provider azure (vEOS) on page 29
• cloud provider aws cloud provider aws (vEOS) on page 29
• proxy proxy (vEOS) on page 33

Interface (Azure)
• active-directory credential email subscription-id active-directory credential email subscription-id (vEOS-Azure) on page 26

Interface (AWS)
• access-key-id access-key-id (vEOS-AWS) on page 25
• region region (vEOS - AWS) on page 34
• secret access-key secret-access_key (vEOS - AWS) on page 35

Cloud Proxy Commands

Global
• cloud proxy cloud proxy (vEOS) on page 30

Interface
• http http (vEOS) on page 30
• https https (vEOS) on page 31
• proxy cloud proxy (vEOS) on page 30

Show Commands

EXEC
• show cloud high-availability show cloud high-availability (vEOS) on page 36
• show cloud high-availability routes show cloud high-availability routes on page 36
• show cloud provider aws show cloud provider aws (vEOS - AWS) on page 37
• show cloud provider azure show cloud provider azure (vEOS - Azure) on page 38
• show cloud proxy show cloud proxy (vEOS) on page 38
**Cloud High Availability CLIs**

The Cloud High Availability CLIs are divided into three separate configuration modes:

- **Cloud Proxy** - For proxy related configuration such as http and https.
- **Cloud Provider** - For cloud provider specific configuration such as region, credential, and proxy name.
- **Cloud High-Availability** - For configurations such as route, next-hop, BFD source interface, and peer.

**access-key-id (vEOS-AWS)**

The cloud provider AWS command places the vEOS in cloud-provider-aws configuration mode. This configuration mode allows user to configure `cloud provider aws access-key-id` command parameters. The `no access-key-id` command removes the configuration from the vEOS `running-config`. The `exit` command returns the vEOS to global configuration mode.

> **Note:** Supported on AWS platform only.

**Command Mode**

Cloud Provider AWS Configuration

**Command Syntax**

`access-key-id(Password_Type)`

`no access-key-id(Password_Type)`

**Parameters**

`Password_Type`

- **0 access-key-id**  The password is a clear-text string. Equivalent to no parameter.
- **7 encrypted_key**  The password is an encrypted string.
- **Text**

**Example:**

The following example configures the AWS access key to encrypted.

```
veos(config)#cloud provider aws
veos(config-cloud-aws)#access-key 0 565656 test
```

**Example:**

The following example removes the AWS access key and returns the vEOS to Global configuration mode.

```
veos(config-cloud-aws)#access-key 0 565656 test
veos(config-cloud-aws)#no access-key 0 565656 test
veos(config)#
```

**Example:**

The following example returns the vEOS to Global configuration mode.

```
veos(config-cloud-aws)#access-key 0 565656 test
veos(config-cloud-aws)#exit
veos(config)#
```
active-directory credential email subscription-id (vEOS-Azure)

The `active-directory credential email subscription-id` command configures Azure's `cloud provider azure active-directory credential` parameters. The `no active-directory` command removes the configuration from the vEOS `running-config`. The `exit` command returns the vEOS to global configuration mode.

**Note:** Supported on Azure platform only.

**Command Mode**

Cloud Provider Azure Configuration

**Command Syntax**

`active-directory credential email subscription-id` ID

`no active-directory credential email subscription-id`

**Parameters**

• `ID` Defines the active directory subscription ID.

**Example:**

The following example places the cloud provider for Azure into the configuration mode.

```
veos(config)#cloud provider azure
veos(config-cloud-azure)#active-directory credential email subscription-id
```

**Example:**

```
veos(config)#cloud provider azure
veos(config-cloud-azure)#active-directory credential email subscription-id
```

azure (vEOS - Azure)

The `azure` command in the `cloud-ha-peer` configuration sub-mode, accessible through the `cloud-ha` configuration mode, allows the user to configure cloud high-availability peer related parameters. The `exit` command returns the vEOS to the to the `cloud-ha-peer` configuration mode.

**Note:** Supported on Azure platform only.

**Command Mode**

Global Cloud High Availability Peer Configuration Submode

**Command Syntax**

`azure`

**Example:**

The following example configures the peer related information for Azure.

```
veos(config)#cloud high-availability
veos(config-cloud-ha)#peer p
veos(config-cloud-ha-peer-veos2)#azure
veos(config-cloud-ha-peer-veos2-azure)#
```

**Example:**
The following example returns the vEOS to the cloud-ha configuration mode.

```
veos(config-cloud-ha-peer-veos2-azure)#exit
veos(config-cloud-ha-peer-veos2)#
```

**backup-gateway (vEOS - Azure)**

The cloud high-availability command in the cloud-ha submode assigns the backup gateway parameters for the Azure high availability peered cloud. The no backup-gateway command removes the configuration from the vEOS running-config. The exit command returns the vEOS to global configuration mode.

**Command Mode**

Cloud HA azure configuration submode

**Command Syntax**

```
backup-gateway [Azure Rt_Info]resource-group[Name]
no backup-gateway [Azure Rt_Info]
```

**Parameters**

- Azure Rt_Info
  - azure-rt-name The azure route name.
  - dest-ip-address/mask The destination IP address.
  - local-ip-address The local IP address.
- resource-group
  - Name Azure resource group name.

**Example:**

The following example configures the parameters for the Azure high availability peered cloud.

```
veos(config)##cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#azure
veos(config-cloud-ha-peer-veos2-azure)#backup-gateway Rt1 10.10.1.1/10
1.1.1.1 resource-group test
```

**Example:**

The following example removes the backup-gateway parameters for the Azure high availability peered cloud.

```
veos(config-cloud-ha-peer-veos2-azure)#no backup-gateway Rt1 10.10.1.1/10
```

**bfd source-interface (vEOS)**

The bfd source-interface command in the cloud-ha configuration submode configures BFD source interface parameters for the high availability peer. The no bfd source-interface command removed the BFD configurations from the vEOS running-config.

**Command Mode**

Global Cloud HA peer configuration mode

**Command Syntax**

```
#bfd source-interface [Interface_Type]single-hop
#no bfd source-interface
```
Parameters

- Interface_Type
  - Ethernet Ethernet Port number <1-4>.
  - Loopback Loopback interface <0-1000>.
  - Tunnel Tunnel interface <0-255>.
- Single-hop Single hop BFD. Default is multi-hop.

Example:
The following example configures Ethernet 1 as the source interface for BFD and multi-hop set as the default.

```
veos(config)#cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#bfd source-interface ethernet 1
```

Example:
The following example configures Tunnel 2 as a single hop the source interface for BFD.

```
veos(config)#cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#bfd source-interface tunnel 2 single-hop
```

Example:
The following example removes the BFD configuration.

```
veos(config-cloud-ha-peer-veos2)#no bfd source-interface
```

cloud high-availability (vEOS)
The `cloud high-availability` command places the vEOS in `cloud-ha` configuration mode. This configuration mode allows user to configure cloud high-availability related parameters. The `exit` command returns the switch to global configuration mode.

Command Mode
Global Cloud High Availability Configuration

Command Syntax
ccloud high-availability

Example:
The following example places the vEOS in the cloud high availability configuration mode.

```
veos(config)#cloud high-availability
veos(config-cloud-ha)#
```

cloud high-availability shutdown (vEOS)
The `shutdown` command in the `cloud-ha` configuration mode disables High Availability for virtual EOS instances running in the cloud environment.

Command Mode
Cloud High Availability configuration

Command Syntax
shutdown

Example:
The following example configures the peer and places it in the cloud high availability configuration mode.

```
veos(config)#cloud high-availability
veos(config-cloud-ha)#shutdown
```

**cloud provider aws (vEOS)**

The `cloud provider aws` command places the vEOS in `cloud-provider-aws` configuration mode. This configuration mode allows user to configure `cloud provider aws` command parameters. The `exit` command returns the vEOS to global configuration mode.

*Note:* Supported on AWS platform only.

**Command Mode**

Global Configuration

**Command Syntax**

`cloud provider aws`

**Example:**

The following example places the cloud provider for AWS into the configuration mode.

```
veos(config)#cloud provider aws
veos(config-cloud-aws)#
```

**Example:**

The following example returns to the global configuration mode.

```
veos(config-cloud-aws)#exit
veos(config)#
```

**cloud provider azure (vEOS)**

The `cloud provider azure` command places the vEOS in `cloud-provider-azure` configuration mode. This configuration mode allows user to configure `cloud provider azure` command parameters. The `exit` command returns the vEOS to global configuration mode.

*Note:* Enabled for Azure platform only.

**Command Mode**

Global Configuration

**Command Syntax**

`cloud provider azure`

**Example:**

The following example places the cloud provider for Azure into the configuration mode.

```
veos(config)#cloud provider azure
veos(config-cloud-azure)#
```
cloud proxy (vEOS)

The cloud proxy command places the vEOS in cloud-proxy configuration mode. This configuration mode allows user to configure the cloud proxy command parameters. The no cloud proxy command disables the named proxy and returns the vEOS to global configuration mode.

Command mode
Global Configuration

Command Syntax

cloud proxy proxy_name
no cloud proxy proxy_name

Parameters

proxy_name The proxy name to configure.

Example:
The following example configures the cloud proxy configuration setting for "test".

veos(config)#
veos(config)#cloud proxy test
veos(config-cloud-proxy-test)#

Example:
This command disables the cloud proxy named "test" and returns the vEOS to global configuration mode.

veos(config-cloud-proxy-test)# no cloud proxy test
veos(config)#

http (vEOS)
The http command in the cloud-proxy configuration submode configures the IP, port, username, and password parameters. The no http command removes the configured cloud proxy information for HTTP from the running-config and returns the vEOS to the global configuration mode.

Command mode
Global Cloud Proxy Configuration

Command Syntax

http[PROXY_IP_PORT][username][password]
no http[PROXY_IP_PORT][username][password]

Parameters

• PROXY_IP_PORT Port number to be used for the HTTP server. Options include:
  • proxy-ip IP address used for the HTTPs proxy. Dotted decimal location.
  • proxy_port HTTPS proxy port. Value ranges from 1 to 65535.
• username Name string.
• password Password string.
  • 0 cleartext-passwd Indicates the cleartext password is in clear text. Equivalent to the no parameter case.
  • 7 encrypted_passwd Indicates encrypted password is md5 encrypted.

Example:
The following example configures the cloud proxy IP, port and username and password for HTTP.

```
veos(config)#
veos(config)#cloud proxy test
veos(config-cloud-proxy-test)# http 1.2.3.4 1234 username test password 7 075E731F1A
veos(config-cloud-proxy-test)#
```

**Example:**
The following example removes the configured cloud proxy information for HTTP from the `running-config`.

```
veos(config-cloud-proxy-test)# no http 1.2.3.4 1234 username test password 7 075E731F1A
veos(config-cloud-proxy-test)#
```

### https (vEOS)

The `https` command in the command in the `cloud-proxy` configuration submode configures the IP, port, username and password parameters. The `no https` command removes the configured cloud proxy information for HTTPS from the `running-config` and returns the vEOS to global configuration mode.

**Command mode**

Global Cloud Proxy Configuration

**Command Syntax**

```
https [PROXY_IP_PORT][[username]][password]
o https [PROXY_IP_PORT][[username]][password]
```

**Parameters**

- **PROXY_IP_PORT** Port number to be used for the HTTP server. Options include:
  - `proxy-ip` IP address used for the HTTPs proxy. Dotted decimal location.
  - `proxy_port` HTTPS proxy port. Value ranges from 1 to 65535.

- **username** Name string.
- **password** Password string.
  - `0 cleartext-passwd` Indicates the cleartext password is in clear text. Equivalent to the `no` parameter case.
  - `7 encrypted_passwd` Indicates encrypted password is md5 encrypted.

**Example:**
The following example configures the cloud proxy IP and port for HTTPS.

```
veos(config)#
veos(config)#cloud proxy test
veos(config-cloud-proxy-test)#https 10.3.255.155 8888
```

**Example:**
The following example removes the configured cloud proxy HTTPS information from the `running-config`.

```
veos(config-cloud-proxy-test)#no https 10.3.255.155 8888
veos(config-cloud-proxy-test)#
```

### recovery (cloud HA peer)

The `recovery wait-time` command in the `cloud-ha-peer` configuration submode defines the amount of time, in seconds, to take control of the local route tables after failure recovery.
Command Mode
Cloud HA peer configuration

Command Syntax

recovery wait-time<time-in-secs>
no recovery wait-time<time-in-secs>

Parameters

• wait-time time-in-secs The defined amount of time to take back control of local route tables after failure recovery. Default is 30 seconds.

Example:

The following example configures the recovery wait time to 90 seconds.

```
veos(config)#cloud ha
veos(config-cloud-ha)#p1
veos(config-cloud-ha-p1)#recovery wait-time 90
```

peer (vEOS)

The peer command in the cloud-ha configuration mode identifies which peer to configure by name. The peer command in the cloud-ha configuration submode configures the cloud high-availability resource group peer related parameters. The no peer command removes the configuration from the vEOS running-config. The exit command returns the vEOS to the cloud-ha configuration mode.

Command Mode
Cloud High Availability Configuration

Cloud High Availability Configuration Submode

Command Syntax

peer

peer ip-address

no peer ip-address

Parameters

• Ip-address The peer IP address.

Example:

The following example configures the cloud high availability peer.

```
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#
```

Example:

The following example configures the peer IP address as 10.10.10.149.

```
veos(config)#cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#peer 10.10.10.149
```

Example:
The following example removes the peer IP address from the vEOS `running-config`.

```bash
veos(config)##cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#no peer 10.10.10.149
```

**primary gateway (vEOS - Azure)**

The `primary-gateway` command in the `cloud-ha` submode assigns the primary gateway parameters for the Azure high availability peered cloud. The `no primary-gateway` command removes the configuration from the vEOS `running-config`.

![Note: Supported on Azure platform only.](image)

**Command Mode**

Cloud HA Azure Configuration Submode

**Command Syntax**

```bash
primary-gateway[Azure Rt_Info]resource-group[Name]
```

```bash
no primary-gateway[Azure Rt_Info]
```

**Parameters**

- **Azure Rt_Info**
  - `azure-rt-name` The azure route name.
  - `dest-ip-address/mask` The destination IP address.
  - `local-ip-address` The local IP address.

- **resource-group**
  - `Name` Azure resource group name.

**Example:**

The following example configures the parameters for the Azure high availability peered cloud.

```bash
veos(config)##cloud high-availability
veos(config-cloud-ha)#peer veos2
veos(config-cloud-ha-peer-veos2)#azure
veos(config-cloud-ha-peer-veos2-azure)#primary-gateway Rt1 10.10.1.1/10 1.1.1.1 resource-group test
```

**Example:**

The following example removes the primary-gateway parameters for the Azure high availability peered cloud.

```bash
veos(config-cloud-ha-peer-veos2-azure)#no primary-gateway Rt1 10.10.1.1/10
```

**proxy (vEOS)**

The `proxy` command configures the cloud provider aws proxy. The `no proxy` command removes the configuration from the `running-config`. The `exit` command returns the vEOS to global configuration mode.

![Note: Supported on AWS platform only.](image)

**Command Mode**
Global Cloud AWS Configuration

Command Syntax

`proxy <proxy_name>`
`no proxy <proxy_name>`

Parameters

• `proxy_name` Proxy name to configure.

Example:

The following example configures the Azure cloud proxy named "test".

```
veos(config)#cloud provider aws
veos(config-cloud-aws)#proxy test
```

recovery wait-time (vEOS)

The `recovery wait-time` command in the `cloud-ha` configuration sub-mode allows takes back control of local route tables after failure recovery. The `no recovery wait-time` command removes the configuration from the vEOS `running-config`. Default is set at 30 seconds.

Command Mode

Cloud High Availability peer configuration

Command Syntax

`recovery wait-time <period>`
`no recovery wait-time <period>`
`default recovery wait-time <period>`

Parameters

• `period` The defined amount of time to take back control of local route tables after failure recovery. Default is 30 seconds.

Example:

The following example shows the wait time is configured to 90 seconds.

```
veos(config-cloud-ha-peer1)#recovery wait-time 90
```

Example:

The following example removes the configured the wait time.

```
veos(config-cloud-ha-peer1)#no recovery wait-time
```

Example:

The following example configures the wait time to the default of 30 seconds.

```
veos(config-cloud-ha-peer1)#default recovery wait-time
```

region (vEOS - AWS)

The `cloud provider aws` command places the vEOS in `cloud-provider-aws` configuration mode. This configuration mode allows user to configure AWS `cloud provider region` command parameters. The `no region` command removes the configuration from the vEOS `running-config`. The `exit` command returns the vEOS to global configuration mode.
Note: Supported on AWS platform only.

Command Mode
Global Cloud Provider AWS Configuration

Command Syntax
region aws-region
no region aws-region

Parameters
• region aws-region Specifies the selected region.

Example:
The following example configures the cloud provider AWS region.

veos(config)\#cloud provider aws
veos(config-cloud-aws)\#region us-west-1
veos(config-cloud-aws)\#

Example:
The following example removes the cloud provider AWS region.

veos(config)\#cloud provider aws
veos(config-cloud-aws)\#no region us-west-1
veos(config-cloud-aws)\#

secret-access_key (vEOS - AWS)
The cloud provider aws command places the vEOS in cloud-provider-aws configuration mode. This configuration mode allows user to configure cloud provider aws secret access-key command parameters. The no secret access-key command removes the configuration from the vEOS running-config. The exit command returns the vEOS to global configuration mode.

Note: Supported on AWS platform only.

Command Mode
Global Cloud Provider AWS configuration

Command Syntax
secret access-key Password_Type
no secret access-key Password_Type

Parameters
• 0 access-key-id The password is a clear-text string. Equivalent to no parameter.
• 7 encrypted_key The password is an encrypted string.
• Text

Example:
The following example configures the AWS secret access key.

veos(config)\#cloud provider aws
veos(config-cloud-aws)\#secret access-key 0 565656 test
veos(config-cloud-aws)\#
Example:
The following example removes the secret access key from the vEOS running-config.

```console
veos(config-cloud-aws)#no secret access-key 0 565656 test
veos(config-cloud-aws)#
```

Example:
The following example returns the vEOS to Global configuration mode.

```console
veos(config-cloud-aws)#secret access-key 0 565656 test
veos(config-cloud-aws)#exit
veos(config)#
```

**show cloud high-availability (vEOS)**
The `show cloud high-availability` command displays the high availability configured settings.

**Command Mode**
EXEC

**Command Syntax**
`show cloud high-availability`

**Example**
This command displays details and status of the cloud high-availability configuration.

```console
veos#show cloud high-availability
Cloud HA Configuration:
Peer address : 10.2.201.149
Source interface : Ethernet1
Enabled : True
Failover recovery time : 5
Status : valid
State : ready
Last failover time : never
Last recovery time : never
Last config validation start time : 0:26:08 ago
Last config validation end time : 0:26:06 ago
Failovers : 0
```

**show cloud high-availability routes**
The `show cloud high-availability routes` command displays the configured local or peer route table, destination IP address and local Next Hop Interface.

**Command Mode**
EXEC

**Command Syntax**
`show cloud high-availability routes`

**Example**
The example below displays high availability routes information.

```console
veos(config)#show cloud high-availability routes
Peer Route  Type Route  ID Destination Next Hop   Interface
----------- ----------- -------------- ---------- ------------
veos6       primary     rtb-1dc75679   0.0.0.0/0  eni-e61d95e7
```
show cloud provider aws (vEOS - AWS)

The **show cloud provider aws** command displays cloud provider information for the AWS platform.

**Command Mode**

EXEC

**Command Syntax**

`show cloud provider aws`

**Example**

The following example displays the AWS cloud configuration.

```
veos# show cloud provider aws
Cloud AWS Configuration
Region : us-west-1
Access key ID :
Access secret key :
Proxy : test
```

**Example:**

The following example displays the primary and backup gateway information for the AWS cloud provider.

```
veos# show run section cloud
cloud provider aws
  us-west-1
  proxy test
  !
  cloud high-availability
  no shutdown
  !
  peer vEOS12
    aws
      backup-gateway rtb-40b72d24 0.0.0.0/0 local-cloud-interface eni-26cb1d27
      backup-gateway rtb-17b32973 0.0.0.0/0 local-cloud-interface eni-1589e714
      backup-gateway rtb-54503330 0.0.0.0/0 local-cloud-interface eni-56cf1957
      primary-gateway rtb-a4be24c0 0.0.0.0/0 local-cloud-interface eni-26cb1d27
      primary-gateway rtb-e64b2882 0.0.0.0/0 local-cloud-interface eni-56cf1957
      primary-gateway rtb-63b02a07 0.0.0.0/0 local-cloud-interface eni-1589e714
    peer address 10.2.201.149
    recovery wait-time 5
    bfd source-interface Ethernet1
    !
    cloud proxy test
    https 10.3.255.155 8888
```
show cloud provider azure (vEOS - Azure)

The show cloud provider azure command displays Azure cloud provider information.

Command Mode
EXEC

Command Syntax
show cloud provider azure

Example
The following example displays the Azure cloud configuration.

veos#show cloud provider azure
Cloud Azure Configuration:
Active credentials : SDK authentication credential file
SDK auth credentials file : flash:
Proxy name :
Active directory credentials :

show cloud proxy (vEOS)

The show cloud proxy command displays the specific cloud proxy information for the named proxy.

Command Mode
EXEC

Command Syntax
show cloud proxy proxy_name

Parameters
• proxy_name Identifies the selected proxy by name.

Example
This command displays the proxy information for the proxy named "test".

veos#show cloud proxy
Cloud Proxy Configuration
Proxy name : test
HTTP proxy : test:075E731F1A@1.2.3.4:1234
HTTPS proxy : 10.3.255.155:8888

This command displays the AWS cloud provider information for the proxy named "test".

veos#show run section proxy
cloud provider aws
    region us-west-1
    proxy test
    !
cloud proxy test
    http 1.2.3.4 1234 username test password 7 075E731F1A
    https 10.3.255.155 8888
Using vEOS Router on the AWS Platform

The vEOS Router, based on the Arista EOS, runs as a virtual machine instance on AWS EC2. Use the vEOS Router to create the various types of virtual machine router instances for AWS deployment, for example, gateway routers and transit routers.

vEOS Router Image Updates

The process to update vEOS Router images is the standard update process used for EOS images. For details on the steps to use, refer to the Arista EOS User Manual (see https://www.arista.com/en/support/product-documentation).

Amazon Machine Image (AMI) Specifications

The AMI provided by Arista utilizes the architecture, type of root device, virtualization type, and interface type required to configure the vEOS Router for a robust AWS deployment.

The specifications of the Arista AMI are:

- **Architecture**: x86_64
- **Virtualization type**: HVM
- **Root Device Type**: EBS
- **Network Interface type**: SR-IOV, ENA (Elastic Network Adapter)

Supported Instance Types

When launching vEOS Router instances, select an AWS instance type. Make sure to select an instance type that meets the resource requirements for the vEOS Router instance that is launching.

The following lists the AWS instance types supported by vEOS Router:

- C4.large
- C4.xlarge
- C4.2xlarge
- R4.large
- R4.xlarge
Methods for Launching vEOS Router Instances

The vEOS Router supports the use of various methods for launching router instances needed in a typical AWS deployment.

The supported methods are:

• Launching vEOS Router Instances Using AWS CloudFormation on page 40
• Launching vEOS Router Instances Using EC2 AWS Marketplace on page 44

Launching vEOS Router Instances Using AWS CloudFormation

Using AWS CloudFormation to launch vEOS Router instances involves creating a CloudFormation stack to use to launch the instance. The created stack provides the base configuration for the instance. As part of this task, select a stack template, which defines the base configuration of the instance.

Make sure to select the stack template that provides the resources required for the instances that are launching. Templates can be obtained from https://github.com/aristanetworks. For more information about AWS CloudFormation stacks and using stack templates, refer to the AWS documentation (see https://aws.amazon.com/documentation/cloudformation/).

Complete these steps to launch vEOS Router instances using AWS CloudFormation.

1. Log in to the Amazon Management Console.
2. Choose Services > CloudFormation.
   The CloudFormation page appears showing the current stacks available to use.
3. Click on the Create Stack button.
   The page refreshes to show the templates that are available to use to create a new stack.

4. Select a nic template for upload, and then click on the Next button.
   Note: Templates can be found in the docs directory. Select the appropriate AMI for launching.
   The page refreshes showing the options for specifying the details for the stack.
5. Enter the Stack Name, Subnet IP Block for each interface, VPC ID, KeyPair Name, UserData in base64 format, AMI ID. (To convert UserData from text to base64 format, use a base64 command on MacOS or Linux machine.)

```
%EOS-STARTUP-CONFIG-START%
hostname myhost
%EOS-STARTUP-CONFIG-END%
<Press CTRL+D>
JUVPUy1TVFSSVQUNPTkJRy1TVFSSVCKaG9zdG5hbWUgbXlob3N0CiVFT1MtU1RBVRVUC1DT05GSUctRU5EJQo=
```

6. Review the details and make changes if needed.
7. Click the **Create** button to create the stack.
8. Wait for the stack creation to complete. Resources created as part of the stack creation process can be viewed in the Resource tab.

9. Click on the vEOS Router instance ID to view the status of vEOS Router instance. The instance ID is shown in the Physical ID column of the Resources tab.
Recommended Usage

AWS cannot auto-assign a public IPv4 address if an EC2 instance is launched or started from the stopped state with multiple network interfaces attached to it. In such cases, the user cannot connect to the instance over IPv4 unless an Elastic IP address is assigned to the primary network interface (eth0). If the user does not want to associate an Elastic IP address with the vEOS Router instance, then it is recommended to attach any additional interface only when the instance is in running state and never to stop and start your instance from thereon. The user may reboot the instance either from AWS console or from within vEOS Router using the CLI or bash commands because the instance reboot does not cause the public IPv4 address to be released as opposed to instance stop. To associate Elastic IP address to your instance or primary network interface, refer to https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html

Launching vEOS Router Instances Using EC2 AWS Marketplace

Launching vEOS Router instances using the EC2 AWS Marketplace gives the ability to create and configure vEOS Router instances in the VPCs of your AWS deployment. This method utilizes Amazon Machine Images (AMIs) to configure the operating system of the instance. Obtain the AMI needed for the instance from the AWS Marketplace. This task involves creating an EC2 key pair, selecting the AMI to configure the operating system of the instance, selecting the instance type, and if needed, configuring advanced details (options) for the instance.

Available Options

During this configuration procedure, choose to configure some options to take advantage of certain features. These optional configuration items are:

• Assigning an IAM role to the instance

To enable AWS services on the instance (for example, AWS CloudWatch logs) assign an IAM role to the instance during this procedure. Assign an IAM role to the instance by:

• Selecting an existing IAM role.
• Creating a new IAM role (an option is provided as part of the procedure to create a new IAM role).

Refer to the following AWS documentation for details about creating EC2 key pairs and creating IAM roles:

• Creating an IAM role ([https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/QuickStartEC2Instance.html](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/QuickStartEC2Instance.html)).

• Using instance user-data to configure the instance

vEOS supports the use of vEOS Router instance user-data to configure vEOS Router instances at launch. This involves uploading instance user-data to the instance by way of the Advanced Details dialog. There is an option of copying and pasting a configuration into the dialog or attaching a configuration file.

For details on composing user data for vEOS Router, see *Using User-data for Configuration of Entities and vEOS Router Instances* on page 57.

Complete the following steps to launch a vEOS Router instances.

1. Log in to the Amazon Management Console.
2. Create an EC2 key pair and download the *pem* file that contains the private key. (The *pem* file may download automatically.)


3. Go to the EC2 Dashboard.

4. From the EC2 Dashboard, click **Instances** in the left pane.

   The Launch Instance page appears.
5. Click on the **Launch Instance** button.
   The page appears for you to select an AMI.

6. Click on **AWS Marketplace** in the left pane.
   Search for **Arista vEOS Router** in the search field to bring up the available vEOS AMIs to use. Select the appropriate AMI for launching.
7. A screen appears showing the user highlights, pricing details and instance types available. Press the Continue button to advance.

8. Click in the left pane.

The Choose an Instance Type page appears.
9. Select an instance type that meets the requirements for the vEOS Router instance. The supported instance types are:
   - C4.large
   - C4.xlarge
   - C4.2xlarge
   - R4.large
   - R4.xlarge
   - R4.2xlarge
   - R4.4xlarge
   - T2 small
   - T2 medium

10. Click on the **Next: Configure Instance Details** button (lower right part of the page).
    The Configure Instance Details page appears.
11. (Optional) Create a new IAM role or select an existing IAM role. (This is required to enable AWS services on the instance, for example, AWS CloudWatch logs.)

12. (Optional) To configure advanced details for the instance, scroll down to the bottom of the page and click on the **Advanced Details** button.

The Advanced Details dialog appears. You use the dialog to upload user-data to configure the instance.

Do one of the following to configure the instance using user-data:

- Choose the **Text** option, and then copy-and-paste **startup-config** in the text box.
- Attach the configuration as a file by **clicking on the file**, and then choose the configuration file to be uploaded.

For details on composing user data for vEOS Router, see *Using User-data for Configuration of Entities and vEOS Router Instances* on page 57.

13. From the Configure Instance Details page, click the **Review and Launch** button.

The Review Instance Launch page appears.
14. Click on the **Launch** button.

A dialog appears for selecting a key pair.

15. Using the **Select a key pair** menu, select the key pair created earlier in the procedure. In this example, the key pair is named "systest."

16. Select the acknowledgment (near the bottom of the dialog), and then click on the **Launch Instances** button. The Launch Status page appears showing the status of the instance.
17. Click on the blue link to the instance to view details about the instance. (The link is in the "Your instances are now launching" box near the top of the page.) The page shows the details for the instance.

18. Make sure the Instance State shows running. Wait for the status to update to running.

19. (Optional) To use the existing subnet and security group for the instance, record the subnet and security group. This information is required when configuring the network interfaces to be attached to the instance.

20. (Optional) Click on the Connect button near the top of the page. The Connect to Your Instance dialog appears.
21. Connect to the instance using the public or private IP address of the instance. The correct syntax is: `ssh -i <privateKey.pem> ec2-user@10.2.1.180`  
Example:  
```bash
#ssh -i <privateKey.pem> ec2-user@10.2.1.180
```

Complete the networking tasks for the vEOS Router instances in the gateway topology (see Network Configuration Tasks for vEOS Router Instances on page 53).

**Configuring the AWS CloudWatch Logs Agent**

The AWS CloudWatch Logs Agent is the mechanism that publishes vEOS Router logs to AWS CloudWatch. Configuring the AWS CloudWatch Logs Agent ensures that the vEOS Router logs published to AWS CloudWatch conform to the selected requirements. The AWS CloudWatch Logs Agent is packaged with the `awslogs.swix` vEOS extension, which is installed and enabled by default when the vEOS Router instances launch through the AWS Marketplace.

Refer to the “AWS CloudWatch Quick Start Guide” to make sure that the vEOS Router instance has the right credentials for logging in to AWS.

⚠️ **Note:** To manually install or uninstall the `awslogs.swix` vEOS extension, see https://eos.arista.com/packaging-and-installing-eos-extensions/. To obtain the `awslogs.swix` vEOS extension, contact Arista TAC if required.

**Where to find vEOS Router logs**

The location where vEOS Router logs are published to depends on the AWS CloudWatch Logs configuration. By default, the logs are located under CloudWatch, "log group" name `veoslogs`.

**Modifying AWS log configuration**

Modify the AWS log configuration by:

- Editing configuration files under the `/mnt/flash/awslogs/` directory.
- Passing instance user-data. Make sure to use the correct start and end markers, which are:

  ```bash
  %AWS-CONFIG-START%
  ```
vEOS Router log filenames

By default, the hostname of the vEOS Router instance is the filename of all vEOS Router logs for that instance.

Network Configuration Tasks for vEOS Router Instances

Complete additional configuration tasks to ensure that the vEOS Router instances launched have the required networking configuration. The configuration tasks include creating the additional network interfaces required by the topology, attaching the new interfaces to vEOS Router instances, and configuring the route table of the AWS Specific Cloud Router.

Creating the Additional Network Interfaces

Creating the additional network interfaces required for the topology ensures that there are interfaces available to attach to vEOS Router instances. When creating the new network interfaces, there is the option of using the subnet and security groups that were automatically assigned to the instance, or specify a different subnet and security groups for the instance.

Pre-requisites:

To use the existing subnet and security group for the vEOS Router instance, make sure to have the following information:

• Subnet ID
• Names of the security groups

Obtain this information by viewing the instance details.

Procedure

Complete these steps to create network interfaces.

1. Go to the EC2 Dashboard.
2. In the NETWORK & SECURITY menu on the left part of the page, select Network Interfaces.
   The page refreshes to show all of the current network interfaces.
3. Select the **Create Network Interface** button.

   The **Create Network Interface** dialog appears.

4. Do the following:
   a) Enter a **description** for the network interface.
   b) Select the **subnet** for the network interface. (This can be the existing subnet for the vEOS Router instance or a different subnet.)
   c) Type the **names of the security groups** for the network interface. (Specify the existing security groups for the vEOS Router instance, or different security groups.)

5. Select the **Yes, Create** button.

   The new network interface is added to the list of interfaces on the page.

6. Repeat steps 3 through 5 to create additional interfaces as needed.

7. For each network interface created, complete steps **a** and **b**:
   a) Select the interface, then choose **Actions > Change Source/Dest Check**.

   The **Change Source/Dest Check** dialog appears showing the selected name of the network interface.
b) Select the **Disabled** option, then click on the **Save** button.

Attach the new network interface to a vEOS Router instance (see *Attaching the New Network Interfaces to Instances* on page 55).

**Attaching the New Network Interfaces to Instances**

Attaching the new network interfaces to vEOS Router instances is the second networking configuration task. This task involves selecting the new network interfaces created in the previous procedure and then attaching the interfaces to vEOS Router instances.

Complete these steps to attach the new network interfaces to vEOS Router instances.

1. Go to the EC2 Dashboard.
2. Open the INSTANCES menu on the left side of the page, then click **Instances**. The page lists all of the current network interfaces.
3. Select the vEOS Router instance to attach a newly created network interface.
   The Attach Network Interface dialog appears.

5. Using the Network Interface menu, select the new network interface created to attach to the instance.
6. Click the Attach button.
7. Use the show interfaces command on the vEOS Router instance to view the new network interfaces created.
   **Example**

```
veos#show interfaces
Ethernet1 is up, line protocol is up (connected)
  Hardware is Ethernet, address is 0235.4079.d2a8 (bia 0235.4079.d2a8)
  Ethernet mtu 8973 bytes, BW 10000000 kbit
  Full-duplex, 10Gb/s, auto negotiation: off, uni-link: n/a
  Up 20 minutes, 42 seconds
  [...]  
Ethernet2 is up, line protocol is up (connected)
  Hardware is Ethernet, address is 0287.4ba7.1f88 (bia 0287.4ba7.1f88)
  Ethernet mtu 8973 bytes, BW 10000000 kbit
  Full-duplex, 10Gb/s, auto negotiation: off, uni-link: n/a
  Up 20 minutes, 42 seconds
```
8. Repeat steps 1 through 7 as needed to attach new network interfaces to instances.

Configure the route table of the AWS Router (see Configuring the Route Table of the AWS Router on page 56).

**Configuring the Route Table of the AWS Router**

To take advantage of the advanced services provided by vEOS, configure the route table of the AWS Router so that traffic is forwarded from the AWS Router to vEOS Router instances. This task involves logging into the AWS Router and modifying route table entries for the vEOS Router instances to which you want traffic forwarded.

Complete these steps to configure the route table of the AWS router.

1. Log in to the AWS Router.
2. Select the network interface that is attached to a vEOS Router instance.
3. Obtain the Subnet ID and the route table ID that corresponds to the subnet in which the vEOS Router instance resides.
   **Example:**
   Subnet ID (subnet-1c68b744).
   Route table ID (rtb-934cf9f7).

4. Edit the route table entry so that it points to the corresponding interface of the vEOS Router in that subnet.
   **Example**
   To reach any subnet other than 10.2.0.0/24, enter the **Target** to be the network interface ID of the locally connected interface of the vEOS Router.

5. (Optional) Repeat steps 2 through 4 to modify route table entries for additional vEOS Router instances.

Configure the AWS CloudWatch Logs Agent (see *Configuring the AWS CloudWatch Logs Agent* on page 52). Configuring the Agent ensures that the vEOS Router logs publish to AWS.

**Using User-data for Configuration of Entities and vEOS Router Instances**

vEOS supports configuration of startup-configuration, AWS CloudWatch, and Cloud HA through the use of user-data. Because user-data can be used to pass in configurations; administrators can take advantage of this feature to quickly configure vEOS Router instances, AWS CloudWatch, and Cloud HA.

**Note:** It is recommended to test vEOS Router configurations on a vEOS Router or EOS device before using them to deploy a new vEOS Router.

**Requirements for Uploading User-data**

To ensure that the user-data is accepted on upload, make sure the user-data meets the following requirements:

- The configuration must be separated by start and end markers.
- Markers are required at the beginning of the line.
- You must upload either text or configuration files (these are the types of files supported by vEOS Router).

EOS configuration for all interfaces can be passed in during deployment. The configuration takes effect as new interfaces attach to the vEOS Router.

**List of Start and End Markers to Use**
This table lists the start and end markers to use when configuring the EOS, AWS, Cloudwatch, and Cloud HA entities. For each specific entity, the configuration file and the location (file path) of the configuration file are given.

<table>
<thead>
<tr>
<th>Entity / Configuration File / Use</th>
<th>Markers</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity: EOS</td>
<td>%EOS-STARTUP-CONFIG-START%</td>
<td>N/A</td>
</tr>
<tr>
<td>File: EOS CLI configuration file</td>
<td>%EOS-STARTUP-CONFIG-END%</td>
<td></td>
</tr>
<tr>
<td>Use: Configure vEOS Router</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%AWS-CONFIG-START%</td>
<td>/mnt/flash/awslogs/aws.conf</td>
</tr>
<tr>
<td></td>
<td>%AWS-CONFIG-END%</td>
<td></td>
</tr>
<tr>
<td>Entity: AWS Logs</td>
<td>%AWSLOGS-CONFIG-START%</td>
<td>/mnt/flash/awslogs/awsconf.conf</td>
</tr>
<tr>
<td>File: aws.conf</td>
<td>%AWSLOGS-CONFIG-END%</td>
<td></td>
</tr>
<tr>
<td>Use: Configure logging parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%AWS-PROXY-START%</td>
<td>/mnt/flash/awslogs/proxy.conf</td>
</tr>
<tr>
<td></td>
<td>%AWS-PROXY-END%</td>
<td></td>
</tr>
<tr>
<td>Entity: AWS Logs</td>
<td>%CLOUDHA-CONFIG-START%</td>
<td>/mnt/flash/cloud_ha_config.json</td>
</tr>
<tr>
<td>File: cloud_ha_config.json</td>
<td>%CLOUDHA-CONFIG-END%</td>
<td></td>
</tr>
<tr>
<td>Use: Configure vEOS Router for High Availability</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sample Instance User-data**
The following sample user-data contains lines to startup the instance and to configure various entities.

The sample contains lines to configure:
- AWS CloudWatch logs (for the us-east-1 region)
- AWS logging parameters
- AWS proxy settings

**Sample**

```
%EOS-STARTUP-CONFIG-START%
! EOS startup config
hostname my-veos
username admin nopassword
username admin sshkey file flash:key.pub
%EOS-STARTUP-CONFIG-END%

%AWS-CONFIG-START%
[plugins]
cwlogs = cwlogs
[default]
region = us-east-1
%AWS-CONFIG-END%
```
%AWSLOGS-CONFIG-END%
[/var/log/messages]
datetime_format = %b %d %H:%M:%S
file = /var/log/messages
buffer_duration = 5000
log_group_name = veoslogs
log_stream_name = {hostname}
initial_position = start_of_file
%AWSLOGS-CONFIG-END%

%AWS-PROXY-START%
HTTP_PROXY=http://<your_proxy>:<your_proxy_port>
HTTPS_PROXY=http://<your_proxy>:<your_proxy_port>
NO_PROXY=169.254.169.254
%AWS-PROXY-END%
Chapter 5

Using the vEOS Router on Microsoft Azure

The vEOS Router, which is based on the Arista EOS, runs as a virtual machine instance on Azure. Use the vEOS Router to create the various types of virtual machine router instances you need for your Azure deployment. For example, gateway routers and transit routers.

vEOS Router Image Updates

The process you use to update vEOS Router images is the standard update process used for EOS images. For details on the steps to use, refer to the Arista EOS User Manual (see https://www.arista.com/en/support/product-documentation).

System Requirements

Describes the vEOS Router Azure minimum support requirements.

vEOS Router Azure instance supports the following instance types:

- **D2_v3** with 2 cores, 8.0GiB RAM, 2 NICs (1,000 Mbps), and a 4GB OS disk.
- **D4_v3** with 4 cores, 16.0GiB RAM, 2 NICs (2,000 Mbps), and a 4GB OS disk.
- **D8_v3** with 8 cores, 32.0GiB RAM, 4 NICs (4,000 Mbps), and a 4GB OS disk.
- **D16_v3** with 16 cores, 64.0GiB RAM, 8 NICs (8,000 Mbps), and a 4GB OS disk.
- **D32_v3** with 32 cores, 128.0GiB RAM, 8 NICs (16,000 Mbps), and a 4GB OS disk.
- **D64_v3** with 64 cores, 256.0GiB RAM, 8 NICs (30,000 Mbps), and a 4GB OS disk.

Launching vEOS Router Azure Instance

There are two methods which can be used to launch a vEOS Router instance.

Below is a summary of each method.

- **Portal Marketplace** This method launches an instance using the Azure Portal Marketplace UI.
- **Azure CLI 2.0**: This method launches an instance using a custom template through the Azure CLI 2.0. The primary advantage of a CLI deployment is the ability to include custom-data and customize your deployment.
Do not deploy the same template twice into a single resource group, because this creates name conflicts. To deploy multiple instances into the same resource group, modify the template, so all resources are renamed, and all IP addresses are unique.

**Creating an Instance using the Portal Marketplace**

To create an instance using the Portal Marketplace, complete the following steps.

1. In the Azure portal, select the green '+' button in the top left of the screen.
2. In the search bar, type "Arista" and press enter.

![Figure 2: Type "Arista"](image)

3. Select the Arista offer you are interested in.
4. Select “Create”.

5. Fill out the required information and press "OK".
6. Configure the VNet and press "OK".

7. Configure the subnets and press "OK".
8. Verify the information is correct and press "OK".

9. Read the Terms and Conditions, then press "Purchase".
Creating an Instance under Azure CLI 2.0

To create an instance under Azure CLI 2.0, complete the following steps.

2. Run `az login` and follow the prompts to authorize the machine.
3. Download the template and parameters files from the GitHub repository. ([https://github.com/Azure/azure-quickstart-templates](https://github.com/Azure/azure-quickstart-templates))
4. Open `<prefix>-parameters.json`. Locate the `<prefix>-single_line_json.sh` user_data.txt script.
5. Copy and paste the generated output into the `customData` value field of the JSON parameters file.
6. Use the script as in the following example:

   ```bash
   #!/usr/bin/bash
   cat $1 | python -c 'import json, sys; print( json.dumps( sys.stdin.read() ) )'
   ```

7. Use the template and parameters JSON files to launch a vEOS Router instance in Azure using the Azure CLI 2.0.

   ```bash
   $ az group create --name ExampleGroup --location "Central US"
   ```

   **Note:** You must use the same location as the storage account where the VHD image is uploaded.

   ```bash
   $ az group deployment create \
   --name ExampleDeployment \
   --resource-group ExampleGroup \
   --template-file <prefix>-template.json \
   --parameters @<prefix>-parameters.json
   ```
**Note:** If you are using a newer version of the Azure CLI 2.0, you may encounter a parameter file parsing bug. To fix this, remove the `@` symbol before the parameters filename.

### Logging into Instance

To log into an instance, complete the following steps.

1. Select the resource group containing your vEOS Router deployment from the **Resource groups** list.
2. Select the item **publicIP**.

![Figure 10: Selecting the PublicIP](image)

3. Locate the IP address and DNS name found on the **Overview** page.
Figure 11: Locating the IP address and DNS

NOTE: If either of these fields is not populated, your instance still deploys. Refresh the page after a couple of minutes.

4. Secure Shell (SSH) to your Virtual Machine (VM) using the IP address or Domain Name Server (DNS) name found in the previous step, using the credentials you gave when you initially setup the VM.

bash# ssh myusername@123.123.123.1
Password: *********

NOTE: It may take between 5-10 minutes for the instance to become reachable after the deployment starts. Refer to the section Troubleshooting Instance on page 69 for additional information.

vEOS Router Startup-Configuration using Instance Custom-Data

Describes launch employing custom-data information.

During the initial launching of the vEOS Router Instance, Azure provides a feature to upload custom-data. The administrator can upload vEOS Router configuration using custom-data at the time of the launching of the vEOS Router Instance.

Custom-data can be used to pass in configuration for multiple entities. Currently, only the EOS configuration is supported in Azure. This configuration must be separated by start and end markers.
Note the following regarding the custom-data:

- Markers must be at the beginning of the line.
- The user is expected to have tested the configurations on a live system before using the configurations to deploy the new vEOS Router. Mis-configuration may result in an unrecoverable instance.
- EOS configuration for all interfaces can be passed in during deployment. The configuration takes effect as the new instances attach to the vEOS Router.

### Sample Instance Custom-Data

Illustrates a sample Instance with custom-data.

```bash
%EOS-STARTUP-CONFIG-START%
! EOS startup config
username admin nopassword
username admin sshkey file flash:key.pub
%EOS-STARTUP-CONFIG-END%
```

### Providing Startup-Configuration using Azure Custom-Data

Adding custom-data to an instance.

Currently, custom-data can only be used on instances deployed using the Azure CLI 2.0.

In order to add custom-data to an instance, the custom-data must be provided as a single-line value with ' \n' delimiting newlines.

Use the `single_line_json.sh` script to convert your custom-data into this format.

```bash
#!/usr/bin/bash
cat $1 | python -c 'import json, sys; print( json.dumps( sys.stdin.read() ) )'
```

Usage of the script is as follows:

```
./single_line_json.sh user_data.txt
```

Copy and paste the generated output into the `customData` value field of the JSON parameters file.

### Troubleshooting Instance

To troubleshoot the instance, complete the following steps.

1. Select the resource group containing your vEOS Router deployment from the Resource groups list.
2. Select the item vEOS Router.
3. Note the status of the VM. It should either be "Creating", "Starting", or "Running".

4. Check the boot diagnostics for any error messages or warnings.
Figure 14: Error messages and warnings

Resources

Additional resources.

1. How To: Deploy Azure Virtual Machines With An Azure Resource Manager (ARM) Template - https://www.youtube.com/watch?v=wi74jR0MRLg
Server Requirements

Server
A server can be either a hardware or software entity.

A hardware server is the physical computer that executes the virtual machine manager or hypervisor and all the virtual machines, also known as the host machine.

A software server is the hypervisor or virtual machine manager that hosts and manages the virtual machines. It is also sometimes referred to as the host.

VMware ESXi Minimum Server Requirements
x86-64 Server class CPU (32-bit CPUs are not supported) with
- Ethernet NICs must be SR-IOV capable
- BIOS / System Firmware support for SR-IOV
- 8 GB free disk space
- 16 GB RAM
- 4 cores running a minimum 2.4GHz or greater and 16 GB memory
- Intel VT-x and VT-d support

VMware ESXi SR-IOV based deployment
- Ethernet NICs must be SR-IOV capable
- BIOS / System Firmware support for SR-IOV

KVM Requirements
vEOS is must be deployed on an x86-64 architecture server running KVM hypervisor.

KVM Minimum Server Requirements
8 GB free disk space
16 GB RAM
x86-64 Server class CPU (32-bit CPUs are not supported) with
- Intel VT-x or AMD-V support for CPU Virtualization
- Intel VT-d or AMD-IOMMU support for PCIe passthrough
- Intel AES-NI support
- 4 CPU cores running at 2.4GHz.

KVM SR-IOV Based Deployment
- Ethernet NICs must be SR-IOV capable
• BIOS / System Firmware support for SR-IOV

**Supported Topologies**

The following scenarios are described in the Hypervisor Chapter

• Launching ESXi using vSphere Web Client
• Launching vEOS on KVM with Linux bridge
• Launching vEOS on KVM with SR-IOV
• Launching vEOS on KVM with PCI-Passthrough

**VMware ESXi Hypervisor**

Describes the launch sequence for VMware ESXi 6.0 and 6.5.

**Launching VMware EXSi 6.0 and 6.5**

How to launch VMware ESXi 6 and ESXi 6.5 for vEOS.

There are different ESXi user interfaces for managing the ESXi host, such as the vSphere Web Client and the ESXi Web Client. The following task is required to launch VMware 6.0 and 6.5 and provides a general guideline on the steps involved in deploying virtual machines with an OVF/OVA template.

**Note:** Arista support suggests using only the Vsphere Web client. The ESXi Web Client may have untested issues.

1. From the vCenter Server WEB-UI navigator, select **Deploy OVF template**.

2. Select the OVA file from the local machine.
3. Select the name and location for vEOS deployment.

4. Select the host, cluster, resource pool or VAPP.
5. Verify the template details.

6. Select **Thick provision eager zeroed** from the datastore.
7. Select the default network.

8. Complete the launch process.
9. Under the **Recent Tasks** tab at the bottom of the page, the progress of deployment displays. Once the deployment is complete, power-on the machine.

---

**Enabling SR-IOV or PCI Passthrough on ESXi**

Describes how to enable single route input/output vitalization (SR-IOV) or PCI passthrough on VMware ESXi.

To enable SR-IOV or PCI passthrough on ESXi, complete the following steps.

1. Navigate to the ESXi host's **Manage**, then select the **Hardware** tab.
2. Locate and select your PIC device/NIC.
3. Use either the **Toggle passthrough** or the **Configure SR-IOV** selection to activate the mode.

4. Reboot the ESXi host for the configuration to take effect.
5. After reboot, the NIC reflects the changes. For SR-IOV, new virtual function devices (VF) is created.
6. Edit the VM and select **Add other device**, then select **PIC Device** to create the **New PIC Device** for the VM.

7. Select the **New PIC Device** to use the SR-IOV VF or PIC Passthrough device.
KVM

This section describes the system requirements, installation and configuration procedures for vEOS.

Server

A server can be either a hardware or software entity.

A **hardware server** is the physical computer that executes the virtual machine manager or hypervisor and all the virtual machines. This is also known as the host machine.

A **software server** is the hypervisor or virtual machine manager that hosts and manages the virtual machines. It is also sometimes referred to as the *host*. In this document specifically, the software server is comprised of RedHat Linux with virtualization support (KVM).

System Requirements

KVM (Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions.

The vEOS is part of the Arista EOS that allows it to deploy as a virtual machine image. This document details the system requirements of vEOS on Linux KVM based hypervisors.

**Minimum Server Requirements**

- Intel x86
- Four cores running at 2.4GHz or greater
- 16 GB memory
- Intel VT-d support
- For SR-IOV based deployment, the NICs need to be SR-IOV capable

**Hypervisor support**

- RedHat 7 with virtualization support. See below for virtualization [https://wiki.centos.org/HowTos/KVM](https://wiki.centos.org/HowTos/KVM). Make sure **libvirt** is installed by executing **virsh list** which should return without errors. Python 2.7+ is needed to run the installation script vSphere 5.5 and 6.0.

**Supported Images**

**KVM vEOS image**

File Name: **EOS.qcow2**

Details: Image Hard Disk that contains vEOS. This file can grow as agents in vEOS generates logs/traces, etc.

**Using Libvirt to Manage vEOS VM on KVM**

Libvirt is an open source library which provides vEOS management of Virtual Machines.

Libvirt supports many functions such as creation, update, and deletion and of VMs.

The complete Libvirt command reference can be found at [http://libvirt.org/virshcmdref.html](http://libvirt.org/virshcmdref.html)

**Define a new VM**

Define a domain from an XML file, by using the **virsh define <vm-definition-file.xml >** command. This defines the domain, but it does not start the domain.
The definition file has vm-name, CPU, memory, network connectivity, and a path to the image. The parameters can be found at https://libvirt.org/formatdomain.html. There is a sample vEOS file in the example below.

**Undefine the Inactive Domain**

Undefine the configuration for the inactive domain by using the `virsh undefine <vm-name>` and specifying its domain name.

**Start VM**

Start a previously defined or inactive domain by using the `virsh start <vm-name>` command.

**Stop VM**

Terminate a domain immediately by using the `virsh destroy <vm-name>` command.

**Managing Networks**

The XML definition format for networks is defined at https://libvirt.org/formatnetwork.html. These commands are similar to the VM, but with a prefix ‘net-’:

- The `virsh net-define <network-definition-file.xml>` command.
- The `virsh net-undefine network-name` command removes an inactive virtual network from the libvirt configuration.
- The `virsh start network-name` command manually starts a virtual network that is not running.
- The `virsh destroy network-name` command shuts down a running virtual network.

**Launching vEOS in LinuxBridge Mode**

Use the script SetupLinuxBridge.pyc usage python SetupLinuxBridge.pyc <bridge-name>

Cut and paste the following XML template into a file (veos.xml) and customize the elements that are in bold below.

- `virsh define <veos define file say veos.xml>`
- `virsh start <veos-name>`
- `virsh console <veos-name>`

```
<domain type='kvm'>
  <!-- veos name, cpu and memory settings -->
  <name>kvs1-veos1</name>
  <memory unit='MiB'>4096</memory>
  <currentMemory unit='MiB'>4096</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <cpu mode='host-model'/>
  <os>
    <type arch='x86_64'>hvm</type>
    <boot dev='cdrom'/>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
</domain>
```
Cut and paste the more interface elements for more interfaces but increment the slot number.

Note that **brWAN** and **brLAN** bridges need to be created beforehand --

```xml
<interface type='bridge'>
  <source bridge='brWAN'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='5' function='0'/>
</interface>

<interface type='bridge'>
  <source bridge='brLAN'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='6' function='0'/>
</interface>
</devices>
</domain>
```

`!-- veos name, cpu and memory settings -->
```xml
<name>kvs1-veos1</name>
<memory unit='MiB'>4096</memory>
<currentMemory unit='MiB'>4096</currentMemory>
<resource>
  <partition>/machine</partition>
</resource>
<cpu mode='host-model'/>
<os>
  <type arch='x86_64'>hvm</type>
  <boot dev='cdrom'/>
  <boot dev='hd'/>
</os>
<features>
  <acpi/>
  <apic/>
  <pae/>
</features>
<clock offset='utc'/>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<devices>
  <emulator>/usr/bin/qemu-system-x86_64</emulator>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' cache='directsync'/>
    <source file='/path-to-veos-image/EOS.qcow2'/>
    <target dev='hda' bus='ide'/>
  </disk>
  <disk type='file' device='cdrom'>
    <driver name='qemu' type='raw'/>
    <source file='/path-to-aboot-image/Aboot-veos-serial.iso'/>
    <target dev='hdc' bus='ide'/>
  </disk>
  <readonly/>
  <alias name='ide0-0-0'/>
  <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
```
<controller type='usb' index='0'>
<alias name='usb0'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0'/>
</controller>

<controller type='pci' index='0' model='pci-root'>
<alias name='pci0'/>
</controller>

<controller type='ide' index='0'>
<alias name='ide0'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0'/>
</controller>

<!-- In this case management is connected to Linux bridge -->

<interface type='bridge'>
<source bridge='brMgmt'/>
<model type='virtio'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x03'/>
</interface>

<serial type='pty'>
<source path='/dev/pts/4'/>
<target port='0'/>
<alias name='serial0'/>
</serial>

<console type='pty' tty='/dev/pts/4'>
<source path='/dev/pts/4'/>
<target type='serial' port='0'/>
<alias name='serial0'/>
</console>

<input type='mouse' bus='ps2'/>
<graphics type='vnc' port='5903' autoport='yes' listen='127.0.0.1'/>
<listen type='address' address='127.0.0.1'/>
</graphics>

<video>
<model type='cirrus' vram='9216' heads='1'/>
<alias name='video0'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x02'/>
</video>

<memballoon model='virtio'>
<alias name='balloon0'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x04'/>
</memballoon>

<!-- Has two data ports on different vlans
Cut and paste the more interface elements for more interfaces but increment the slot number.
Note that brWAN and brLAN bridges need to be created beforehand -->

<interface type='bridge'>
<source bridge='brWAN'/>
<model type='virtio'/>
<address type='pci' domain='0x0000' bus='0x00' slot='5'/>
</interface>

<interface type='bridge'>
<source bridge='brLAN'/>
<model type='virtio'/>
<address type='pci' domain='0x0000' bus='0x00' slot='6'/>
</interface>

<devices>
</domain>
Example Deployment

VIRTIO & Linux Bridging Deployment

vEOS can employ para-virtualized network I/O interfaces, which in Linux KVM is also known as Virtio. Each NIC is connected to a unique underlying Linux layer-2 bridge in the hypervisor which in-turn provides access to an uplink.

In this example,

• Ethernet1 connects to the physical Ethernet port that connects to the WAN through a LinuxBridge. The Router is configured with a WAN IP address on this port.
• Ethernet2 connects to the physical ethernet port that connects to the LAN through a LinuxBridge.
• Server IP address in the diagram is assumed to be configured on the LAN LinuxBridge device.

Note: Arista recommends using Ethernet1 for WAN and Ethernet2 for LAN. However, any vEOS port can be used.

Linux Bridge and Virtio-based Deployment

![Diagram of Linux Bridge and Virtio-based Deployment]

Figure 15: Linux Bridge and Virtio-based Deployment
Setting Up the Host for Single Root I/O Virtualization (SR-IOV)

Single Root I/O Virtualization (SR-IOV) allows a single PCIe physical device under a single root port to appear to be multiple physical devices to the hypervisor.

The following tasks are required to set up the host for SR-IOV.

1. Verify the IOMMU Support.

   Use the `virt-host-validate` Linux command to check IOMMU (input/output memory management unit) support. If it does not "PASS" for IOMMU, check the BIOS setting and kernel settings.

   The example below is what should be displayed.

   ```
   [arista@solution]$ virt-host-validate
   QEMU: Checking for device assignment IOMMU support : PASS
   QEMU: Checking if IOMMU is enabled by kernel       : PASS
   ```

2. Verify the Drivers are Supported.

   Ensure the PCI device with SR-IOV capabilities is detected. In the example below, an INTEL 82599 ES network interface card is detected which supports SR-IOV.

   Verify the ports and NIC IDs that are in bold in the `lspci | grep Ethernet` Linux command output below.

   ```
   # lspci | grep Ethernet
   01:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   01:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   01:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   01:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   81:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   81:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   81:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   81:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
   82:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
   82:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
   83:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
   83:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
   ```

3. Verify the driver kernel is active.

   After confirming the device support, the driver kernel module should load automatically by the kernel. To verify the driver kernel is active, use the `lsmod | grep igb` Linux command.

   ```
   [root@kvmssolution]# lsmod | grep igb
   igb              197328  0
   ptp              19231  2 igb,ixgbe
   dca              15130  2 igb,ixgbe
   i2c_algo_bit     13413  2 ast,igb
   ```

Server Requirements
4. Activate Virtual Functions (VFs).

The maximum number of supported virtual functions depends on the type of card. The example below shows that the PF identifier 82:00.0 supports a total of 63 VFs.

```
$ cat/sys/bus/pci/devices/0000:82:00.0/sriov_totalvfs
63
```

To activate the seven VFs per PFs and make them persistent after reboot, add the line options igb max_vfs=7 in ixgbe.conf and the sriov.conf files in /etc/modprobe.d

Use the `rmmod ixgbe` and `modprobe ixgbe` Linux commands to unload and reload the module.

5. Verify the VFs are detected.

Verify the VFs are detected by using the `lspci | grep Ethernet` Linux command. For the two identifiers 82:00.0 and 82:00.1, 14 VFs are detected.

```
# lspci | grep Ethernet
82:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
82:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
82:10.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.1 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.5 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.6 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.7 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.1 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.5 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
```

6. Locate the serial numbers for the PFs and VRFs.

Locate the serial numbers for the PFs and VFs. The Linux `virsh nodedev-list | grep 82` command below displays the serial number for identifiers 82:00.0 and 82:00.1. The first two numbers are the serial numbers for the PFs and the remaining are the serial numbers for the VFs.

```
# virsh nodedev-list | grep 82
pci_0000_82_00_0
```
7. Select the serial number of the VF.

Select the serial number of the VF that will attach to the VM (vEOS). Using the Linux `virsh nodedev-dumpxml <serial number>` command, locate the bus, slot, and function parameters. For example, serial number: `pci_0000_82_11_1` displays the following details.

```
# virsh nodedev-dumpxml pci_0000_82_11_1
<device>
  <name>pci_0000_82_11_1</name>
  <path>/sys/devices/pci0000:80/0000:80:02.0/0000:82:11.1</path>
  <parent>computer</parent>
  <driver>
    <name>ixgbevf</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>130</bus>
    <slot>17</slot>
    <function>1</function>
    <product id='0x10ed'>82599 Ethernet Controller Virtual Function</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
    <phys_function>
      <address domain='0x0000' bus='0x82' slot='0x00' function='0x1'/>
    </phys_function>
    <iommuGroup number='71'>
      <address domain='0x0000' bus='0x82' slot='0x11' function='0x1'/>
    </iommuGroup>
    <numa node='1'/>
    <pci-express>
      <link validity='cap' port='0' width='0'/>
      <link validity='sta' width='0'/>
    </pci-express>
  </capability>
</device>
```

8. Create a new Interface.

Shutdown the vEOS VM if it is already running. Open the XML file for the specific vEOS VM for editing using the Linux command `virsh edit <vm-name>`. In the interface section, create a new interface by adding the details as shown below. The bus, slot, and function values are in the hexadecimal format of the decimal values found in step 7.

```
<interface type='hostdev' managed='yes'>
  <source>
    <address type='pci' domain='0x0000' bus='0x82' slot='0x11' function='0x1'/>
  </source>
</interface>
```
9. Start the vEOS VM. Verify there is an added interface on the VM. Using the command `ethtool -i et9` to verify that the driver for the added interface is `ixgbevf`.

```
veos(config)# show interface status

Port   Name    Status  Vlan  Duplex  Speed  Type  Flags
Et9    notconnect routed unconf 10/100/1000
Ma1 connected   routed  a-full  1G      10/100/1000

[admin@vEOS]$ ethtool -i et9

driver: ixgbevf
version: 2.12.1-k
firmware-version: 0000:00:0c.0
supports-statistics: yes
supports-test: yes
supports-eprom-access: no
supports-register-dump: yes
supports-priv-flags: no
```

**Launching SR-IOV**

vEOS can also use PCIE SRI-OV I/O interfaces. Each SRI-OV NIC is passed-through to the VM such that network I/O does not hit the hypervisor. In this model, the hypervisor and multiple VMs can share the same NIC card.

SR-IOV has the following advantages over LinuxBridge:

- Higher Performance ~ 2x.
- Better latency and jitter characteristics.
- vEOS directly receives physical port state indications from the virtual device.
- Using SR-IOV virtualize the NIC.
- The NICs have a built-in bridge to do basic bridging.
- Avoids software handling of the packets in the kernel.
Setting Up the Host and Launching PCI Pass-through

Set up a networking device to use PCI pass-through.

When sharing resources are not efficient, or packets are consumed by a virtualized switch before reaching the VM (vEOS), implementing PCI Pass-through for NIC provides dedicated and non-filtered network resources to the VM.

1. Identify Available Physical Functions.

Similar to the SR-IOV, identify an available physical function (a NIC in this scenario) and its identifier. Use the `lspci | grep Ethernet` Linux command to display the available physical functions.
In this example, 82:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection is the physical function and 82:00.0 is the device identification code.

```
# lspci | grep Ethernet
01:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
01:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
01:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
01:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
81:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
81:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
81:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
81:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
82:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
82:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
83:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
83:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
```

2. Verify Available Physical Functions.

Verify the available physical functions by using the `virsh` Linux commands.

```
[p_pci_0000_82_00_0]

[p_pci_0000_82_00_0]
```

In this example, the domain is 0 (Hex domain=0x0), the bus is 130 (Hex bus=0x82), the slot is 0 (Hex slot=0x0), and function is 0 (Hex function=0x0).

With the domain, bus, slot, and function information, construct the device entry and add it into the VMs XML configuration.
3. Verify the NIC was detected by the VM.

When starting the VM (vEOS in this case), the VM should detect NIC.

```
veos#bash
Arista Networks EOS shell
[admin@veos ~]$ lspci | grep Ethernet
00:03.0 Ethernet controller: Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection (rev 01)
00:05.0 Ethernet controller: Red Hat, Inc Virtio network device
[admin@veos ~]$
```

4. Verify Driver Requirements.

If the NIC is supported by the vEOS and any other driver requirements are met, the corresponding ethernet interfaces are available to use on the vEOS. Use the `show interface` command to display the available vEOS Ethernet interfaces.

```
veos#show interface status
Port   Name   Status     Vlan  Duplex  Speed  Type      Flags
Et1    connected routed   full  10G      10/100/1000
Ma1    connected routed   a-full a-1G  10/100/1000
```

```
veos#bash
bash-4.3# ethtool -i et1
driver: ixgbe
version: 4.2.1-k
firmware-version: 0x18b30001
bus-info: 0000:00:03.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

**Example Deployment**

vEOS can use passthrough I/O interfaces where the network I/O does not hit the hypervisor. In this model, the VM owns the entire network card, thus fully bypassing the hypervisor.

Setting up SR-IOV is initially more involved. Arista recommends starting out with LinuxBridge.

- SR-IOV has the following advantages over LinuxBridge Higher Performance ~ 2x
- Better latency and jitter characteristics
- vEOS directly receives physical port state indications from the virtual device.
Linux SRIOV PCI Passthrough-based Deployment

Figure 17: Linux SRIOV PCI Passthrough-based Deployment
Chapter 7

IPsec Support

The vEOS Router provides robust support for the use of IPsec to establish and maintain IPsec tunnels for secure or encrypted communications between virtual router peer instances as well as virtual peer instances to non-virtual routers.

The vEOS Router supports the use of IPsec to:

- Secure the communications between vEOS Router instances.
- Secure the communications between vEOS Router instances and third party virtual router instances.

**Note:** For the latest information on the types of virtual routers that can share IPsec tunnels with vEOS Router, see the vEOS Router Release Notes.

- **Supported Tunnel Types** on page 96
  The vEOS Router supports the use of two basic types of IPsec tunnels. The tunnel types are determined based on the encapsulation mode.

- **Requirements when Behind a NAT** on page 96
  The vEOS Router supports the use of NAT-Traversal to communicate with the remote peer virtual router. To ensure that the tunnel configuration between the vEOS Router and peer router is successful, make sure that vEOS Router tunnel configuration meets the requirements for using NAT.

- **Using IPsec on vEOS Router Instances** on page 97
  The vEOS Router enables you to establish and maintain GRE-over-IPsec and VTI IPsec tunnels for secure or encrypted communications between peer vEOS Router instances.

- **Using IPsec on vEOS and Third Party Devices** on page 103
  The vEOS Router enables you to establish and maintain IPsec tunnels for secure or encrypted communications between vEOS Router instances and third party peer router instances.

- **Example of Running-configs for IPsec Tunnels** on page 101
  This example shows the running configurations for a VTI IPsec tunnel between a vEOS Router instance and a Palo Alto firewall VM router instance.

- **IPsec Show Commands** on page 112
  The vEOS Router provides commands you can use to view all current or established IPsec tunnels and to view all profiles currently being used by established tunnels.
Supported Tunnel Types

The vEOS Router supports the use of two basic types of IPsec tunnels. The tunnel types are determined based on the encapsulation mode.

The supported tunnel types are:

**GRE-over-IPsec**
- In GRE-over-IPsec encapsulation mode, the application payload is first encapsulated within a GRE packet. IPsec then encrypts the GRE packet, which results in the packet being encapsulated and encrypted by the IPsec header.
- Select this encapsulation type by specifying `tunnel mode gre` for the tunnel interface to which the IPsec profile is applied. This ensures that the packets forwarded on the interface are encrypted.
- When using GRE-over-IPsec encapsulation mode, both IPsec mode options are supported (select either `transport` or `tunnel`).

**VTI IPsec**
- In VTI encapsulation mode, the application payload is directly encapsulated and encrypted by the IPsec header.
- Select this encapsulation type by specifying `tunnel mode ipsec` for the tunnel interface to which the IPsec profile is applied. This ensures that the packets forwarded on the interface are encrypted.
- When using VTI encapsulation mode, set the IPsec mode to `tunnel`. The `transport` option under the IPsec mode has no effect.

Requirements when Behind a NAT

The vEOS Router supports the use of NAT-Traversal to communicate with the remote peer behind a NAT. Configure the tunnel source with the outgoing interface IP address on the router.

**Flow Parallelization**

To achieve high throughput over an IPsec connection, enable the IPsec flow parallelization feature. When the feature is enabled, multiple cores are used to parallelize the IPsec encryption and decryption processing. To enable this feature, include the `flow parallelization encapsulation udp` command in the IPsec profile configuration.

**Note:** The feature must be enabled on both sides of the tunnel. Other vendors do not support Flow Parallelization.

**Note:** This feature should be used with GRE over IPsec.

If the IPsec session is established without the feature enabled, complete the following tasks:
- Under the IPsec profile for the tunnel use the `flow parallelization encapsulation udp` command to enable the feature.
- Shutdown the tunnel on the tunnel interface.
- Bring the tunnel back up on the tunnel interface. After it is up, this enables the feature.
Using IPsec on vEOS Router Instances

The vEOS Router establishes and maintains GRE-over-IPsec and VTI IPsec tunnels for the secure or encrypted communications between peer vEOS Router instances.

Topology

Use the vEOS Router to establish and maintain IPsec tunnels between peer vEOS Router instances in different topologies of varying complexity.

The diagram below represents a basic IPsec tunnel configuration in which vEOS Router instances are using an IPsec tunnel.

![vEOS Router Instance Using a Basic IPsec Tunnel](image)

Figure 18: vEOS Router Instance Using a Basic IPsec Tunnel

The vEOS Router establishes and maintains IPsec tunnels for secure or encrypted communications between vEOS Router instances and third party devices peer router instances.

The basic process for establishing secure communications using IPsec involves the following tasks:

- Creating IKE Policy for establishing IKE with the peer.
- Specifying the encryption, integrity protocols for the Security Association (SA) Policy.
- Apply IKE and SA policies to a given profile.
- Apply the profile to a tunnel interface.

Configuring IPsec Tunnels on vEOS Router Instances

Use this procedure to configure GRE-over-IPsec or VTI IPsec tunnels on peer vEOS Router instances.

The procedure provides all of the steps required to set up either GRE-over-IPsec or VTI IPsec tunnels. Most of the steps are the same for both tunnel types (steps 1 through 6 are the same). Step 7 is the step to select the tunnel type.

Note: vEOS Router by default uses IKE version 2 for all IPsec tunnels. To configure a tunnel that uses IKE version 1, explicitly configure the vEOS Router to use IKE version 1.

Procedure

Complete the following steps to configure GRE-over-IPsec or VTI IPsec tunnels on vEOS Router instances. This configuration will be the default IKE version 2 procedure.

1. Use this command to enter IP security mode.

```
veos(config)#ip security
```
2. To use IKE version 1, complete the following before completing the default IKE version the steps below.

```
veos(config)#ip security
veos(config-ipsec)#ike policy ike-peerRtr
veos(config-ipsec-ike)#version 1
```

3. Create an IKE Policy to be used to communicate with the peer to establish IKE. You have the option of configuring multiple IKE policies.

The default IKE Policy values are:

- **Encryption - AES256**
- **Integrity - SHA256**
- **DH group - Group 14**
- **IKE lifetime - 8 hours**

**Example:**

```
veos(config-ipsec)#ike policy ike-vrouter
veos(config-ipsec-ike)#encryption aes256
veos(config-ipsec-ike)#integrity sha256
veos(config-ipsec-ike)#dh-group 24
veos(config-ipsec-ike)#version 2
```

4. If the router is behind a NAT, configure the `local-id` with the local public IP address. The public IP corresponds to the underlying interface over which the IKE communications are done with the peer.

**Example:**

```
veos(config-ipsec-ike)#local-id <public ip address>
```

5. Create an IPsec Security Association policy to be used in the data path for encryption and integrity. Use the option of enabling Perfect Forward Secrecy by configuring a DH group to the SA.

**Example:** In this example, **AES256** is used for encryption, **SHA 256** is used for integrity, and Perfect Forward Secrecy is enabled (the **DH group is 14**).

```
veos(config-ipsec-ike)#sa policy sa-vrouter
veos(config-ipsec-sa)#esp encryption aes256
veos(config-ipsec-sa)#esp integrity sha256
veos(config-ipsec-sa)#pfs dh-group 14
veos(config-ipsec-sa)#sa lifetime 2
veos(config-ipsec-sa)#exit
```

6. Bind or associate the IKE and SA policies together using an IPsec profile. Provide a shared-key, which must be common on both peers. The default profile assigns default values for all parameters that are not explicitly configured in the other profiles.

**Example:** In this example, **tunnel mode** is set to **transport**. The IKE Policy **ike-peerRtr** and SA Policy **sa-peerRtr** are applied to profile **peer-Rtr**. Dead Peer Detection is enabled and configured to delete the connection when the peer is down for more than 50 seconds. The peer **peer-Rtr** is set to be the responder.

```
veos(config-ipsec)#profile default
veos(config-ipsec-profile)#ike-policy ikedefault
veos(config-ipsec-profile)#sa-policy sadefault
veos(config-ipsec-profile)#shared-key arista
veos(config-ipsec-profile)#profile vrouter
veos(config-ipsec-profile)#ike-policy ike-vrouter
veos(config-ipsec-profile)#sa-policy sa-vrouter
veos(config-ipsec-profile)#dpd 10 50 clear
veos(config-ipsec-profile)#connection add
veos(config-ipsec-profile)#mode transport
```

7. Configure the WAN interface to be the underlying interface for the tunnel. You must specify an L3 address for the tunnel. If you do not, the vEOS Router cannot route packets using the tunnel.
8. Apply the IPsec profile to a new tunnel interface. You create the new tunnel interface as part of this step. You can choose to configure the tunnel as a GRE-over-IPsec tunnel, or a VTI IPsec tunnel.

**Example (GRE-over-IPsec):** In this example, the new tunnel interface is `Tunnel0`. The new tunnel interface is configured to use IPsec, and the tunnel mode is set to GRE. The other end of the tunnel also needs to be configured as a GRE-over-IPsec tunnel.

```
veos(config)#interface tunnel0
veos(config-if-Tu0)#ip address 1.0.3.1/24
veos(config-if-Tu0)#tunnel mode gre
veos(config-if-Tu0)#mtu 1394
veos(config-if-Tu0)#tunnel source 1.0.0.1
veos(config-if-Tu0)#tunnel destination 1.0.0.2
veos(config-if-Tu0)#tunnel ipsec profile vrouter
```

**Example (VTI IPsec):** To configure a VTI IPsec tunnel, you need to set the tunnel mode to `tunnel mode ipsec`. The other tunnel element settings are the same as the settings for GRE-over-IPsec.

```
veos(config)#interface tunnel0
veos(config-if-Tu0)#ip address 1.0.3.1/24
veos(config-if-Tu0)#tunnel mode ipsec
veos(config-if-Tu0)#mtu 1394
veos(config-if-Tu0)#tunnel source 1.0.0.1
veos(config-if-Tu0)#tunnel destination 1.0.0.2
veos(config-if-Tu0)#tunnel ipsec profile vrouter
```

**Optional Steps**
To move the tunnel interface to a different VRF, complete step 9. To achieve high throughput, complete step 10.

9. Create the GRE-over-IPsec tunnel interface in a VRF using the `vrf forwarding` command. If a VRF is needed, create one then create and configure the GRE tunnel interface. If tunnels in different VRFs need to share the IPsec connection, configure the same tunnel source, destination, IPsec profile, and a unique tunnel key for each tunnel.

**Note:** If tunnels in different VRFs need to share the IPsec connection, specify the same source, destination, and IPsec profile.

**Example:**

```
veos(config)#vrf definition red
veos(config-vrf-red)#rd 1:3
veos(config-vrf-red)#interface tunnel0
veos(config-if-Tu0)#tunnel key 100
veos(config-if-Tu0)#vrf forwarding red
veos(config-if-Tu0)#ip address 1.0.3.1/24
veos(config-if-Tu0)#mtu 1394
veos(config-if-Tu0)#tunnel source 1.0.0.1
veos(config-if-Tu0)#tunnel destination 1.0.0.2
veos(config-if-Tu0)#tunnel ipsec profile vrouter
veos(config)#vrf definition blue
veos(config-vrf-blue)#rd 1:4
veos(config-vrf-blue)#interface tunnel1
veos(config-if-Tu1)#tunnel key 200
```
veos(config-if-Tu1)#vrf forwarding blue
veos(config-if-Tu1)#ip address 1.0.4.1/24
veos(config-if-Tu1)#tunnel mode gre
veos(config-if-Tu1)#mtu 1394
veos(config-if-Tu1)#tunnel source 1.0.0.1
veos(config-if-Tu1)#tunnel destination 1.0.0.2
veos(config-if-Tu1)#tunnel ipsec profile vrouter

10. Enable the IPsec flow parallelization feature to achieve high throughput over the IPsec tunnel. To enable the feature, include the flow parallelization encapsulation udp command in the IPsec profile configuration. Then, apply the IPsec profile configuration to the tunnel interface.

**Example: (IPsec profile configuration)**

veos(config-ipsec)#profile vrouter
veos(config-ipsec-profile)#ike-policy ike-vrouter
veos(config-ipsec-profile)#sa-policy sa-vrouter
veos(config-ipsec-profile)#dpd 10 50 clear
veos(config-ipsec-profile)#connection start
veos(config-ipsec-profile)#mode transport
veos(config-ipsec-profile)#flow parallelization encapsulation udp

**Example: (Applying IPsec profile to tunnel interface)**

veos(config)#interface tunnel0
veos(config-if-Tu0)#tunnel ipsec profile vrouter

**Note:** Repeat step 9 on the other end of the tunnel. The IPsec flow parallelization feature must be enabled on both end of the tunnel.

### Examples of Running-configurations for GRE-over-IPsec Tunnels

The following examples show the running configurations for two vEOS Router instances (vEOS1 and vEOS2). The instances are the tunnel endpoints of a GRE-over-IPsec tunnel.

**Running Configuration for vEOS1**

```plaintext
ip security
ike policy ikebranch1
integrity sha256
dh-group 15
! sa policy sabranch1
  sa lifetime 2
  pfs dh-group 14
! profile hq
  mode tunnel
  ike-policy ikebranch1
  sa-policy sabranch1
  connection add
  shared-key keyAristaHz
  dpd 10 50 clear
! interface Tunnel1
  mtu 1404
  ip address 1.0.3.1/24
  tunnel mode gre
  tunnel source 1.0.0.1
```
Examples of Running-configurations for VTI IPsec Tunnels

The following examples show the running configurations for two vEOS Router instances (vEOS1 and vEOS2). The instances are the tunnel endpoints of a VTI IPsec tunnel.

Running Configuration for vEOS1

```conf
ip security
ike policy ikebranch1
integrity sha256
dh-group 15

ike policy ikebranch2
dh-group 15
version 1
local-id 200.0.0.1

ike policy ikedefault

sa policy sabranch1
sa lifetime 2
pfs dh-group 14

profile hq
mode tunnel
ike-policy ikebranch1
sa-policy sabranch1
connection start
shared-key keyAristaHq
dpd 10 50 clear

interface Tunnel1
mtu 1404
ip address 1.0.3.2/24
tunnel mode gre
tunnel source 1.0.0.1
tunnel destination 1.0.0.1
tunnel ipsec profile hq

interface Ethernet2
no switchport
ip address 1.0.0.2/24
```

Running Configuration for vEOS2

```conf
tunnel destination 1.0.0.2
tunnel ipsec profile hq

interface Ethernet1
no switchport
ip address 1.0.0.1/24

Running Configuration for vEOS2

ip security
ike policy ikebranch1
integrity sha256
dh-group 15

ike policy ikebranch2
dh-group 15
version 1
local-id 200.0.0.1

ike policy ikedefault

sa policy sabranch1
sa lifetime 2
pfs dh-group 14

profile hq
mode tunnel
ike-policy ikebranch1
sa-policy sabranch1
connection start
shared-key keyAristaHq
dpd 10 50 clear

interface Tunnel1
mtu 1404
ip address 1.0.3.2/24
tunnel mode gre
tunnel source 1.0.0.2
tunnel destination 1.0.0.1
tunnel ipsec profile hq

interface Ethernet2
no switchport
ip address 1.0.0.2/24
```
sa lifetime 2
pfs dh-group 14
!
profile hq
mode tunnel
ike-policy ikebranch1
sa-policy sabranch1
connection add
shared-key keyAristaHq
dpd 10 50 clear
!
interface Ethernet1
no switchport
ip address 1.0.0.1/24
!
interface Management1
ip address dhcp
!
interface Tunnel1
mtu 1404
ip address 1.0.3.1/24
tunnel mode ipsec
tunnel source 1.0.0.1
tunnel destination 1.0.0.2
tunnel ipsec profile hq
!

Running Configuration for vEOS2

ip security
ike policy ikebranch1
integrity sha256
dh-group 15
!
ike policy ikebranch2
dh-group 15
version 1
local-id 200.0.0.1
!
ike policy ikedefault
!
sa policy sabranch1
sa lifetime 2
pfs dh-group 14
!
profile hq
mode tunnel
ike-policy ikebranch1
sa-policy sabranch1
connection start
shared-key keyAristaHq
dpd 10 50 clear
!
interface Ethernet2
no switchport
ip address 1.0.0.2/24
!
interface Management1 ip address dhcp
!
interface Tunnel1
mtu 1404
Using IPsec on vEOS and Third Party Devices

The vEOS Router establishes and maintains IPsec tunnels for secure or encrypted communications between vEOS Router instances and third party devices peer router instances.

The basic process for establishing secure communications using IPsec involves these tasks:

• Creating IKE Policy for establishing IKE with the peer.
• Specifying the encryption, integrity protocols for the Security Association (SA) Policy.
• Apply IKE and SA policies to a given profile.
• Apply the profile to a tunnel interface.

Topology

Use the vEOS Router to establish and maintain IPsec tunnels between vEOS Router instances and third party router instances in different topologies of varying complexity.

The following diagram represents a basic IPsec tunnel configuration in where a vEOS Router instance and a third party router instance is connected using an IPsec tunnel.

Figure 19: IPsec Interoperability

Interoperability Support

The vEOS Router establishes and maintains IPsec tunnels for the secure or encrypted communications between vEOS Router instances and third party device peer router instances.

Below lists the types of IPsec tunnels to set up between vEOS Router instances and third party virtual router instances.

• Palo Alto Firewall VM
  • Set up these types of IPsec tunnels between vEOS Router instances and Palo Alto firewall VM router instances.
  • VTI IPsec
• CSR
• Set up these types of IPsec tunnels between vEOS Router instances and CSR router instances.
  • GRE-over-IPsec
  • VTI IPsec

• AWS VPN Specific Cloud
  • Set up these types of IPsec tunnels between vEOS Router instances and AWS VPN Specific Cloud router instances.
    • VTI IPsec

• vSRX
  • Set up these types of IPsec tunnels between vEOS Router instances and vSRX router instances.
    • VTI IPsec

Note: See Supported Tunnel Types on page 96 for descriptions of the GRE-over-IPsec and VTI IPsec tunnel types.

vEOS Routers and Palo Alto firewall AM vEOS Routers and Palo Alto Firewall VM on page 104
vEOS Routers and CSR vEOS Routers and CSR on page 113
vEOS and AWS Specific Cloud vEOS Routers and AWS Specific Cloud Configuration on page 121
vEOS and vSRX vEOS Routers and vSRX on page 104

vEOS Router and Palo Alto Firewall VM

The vEOS Router establishes and maintains IPsec tunnels for secure or encrypted communications between vEOS Router instances and third party device peer router instances.

vEOS Router Configuration

Use this procedure to configure GRE-over-IPsec tunnels on a vEOS Router instance. Once the procedure is complete, configure the other tunnel end-point on the third party peer router.

Note: The vEOS Router by default uses IKE version 2 for all IPsec tunnels. If you want to configure a GRE-over-IPsec tunnel that uses IKE version 1, explicitly configure the vEOS Router to use IKE version 1.

Procedure

Complete the following steps to configure the vEOS Router instance to share a GRE-over IPsec tunnel.

To use IKE version 1, complete the section below, then continue with the following steps. To use the default version IKE version 2, begin with Step 1 below.

veos(config)#ip security
veos(config-ipsec)#ike policy ike-peerRtr
veos(config-ipsec-ike)#version 1

1. Use this command to enter IP security mode.
  veos(config)#ip security

2. Create an IKE Policy used to communicate with the peer to establish IKE Phase 1. There is an option of configuring multiple IKE policies.
   The default IKE Policy values are:
- **Encryption** - AES256
- **Integrity** - SHA256
- **DH group** - Group 14
- **IKE lifetime** - 8 hours

**Example:**

veos(config-ipsec)#ike policy ike-vrouter
veos(config-ipsec-ike)#encryption aes256
veos(config-ipsec-ike)#integrity sha256
veos(config-ipsec-ike)#dh-group 24
veos(config-ipsec-ike)#version 2
veos(config-ipsec-ike)#exit
veos(config-ipsec)#ike policy ike-default
veos(config-ipsec-ike)#version 2
veos(config-ipsec-ike)#exit

3. If the router is behind a NAT, configure the **local-id** with the local public IP address.
   **Example:**

   veos(config-ipsec-ike)#local-id <public ip address>

4. Create an IPsec Security Association policy used in the data path for encryption and integrity. The is an option of enabling Perfect Forward Secrecy by configuring a DH group to the SA.
   **Example:** In this example, **AES256** is used for encryption, **SHA 256** is used for integrity, and Perfect Forward Secrecy is enabled (the DH group is **14**).

   veos(config-ipsec)#sa policy sa-vrouter
   veos(config-ipsec-sa)#esp encryption aes256
   veos(config-ipsec-sa)#esp integrity sha256
   veos(config-ipsec-sa)#pfs dh-group 14
   veos(config-ipsec-sa)#sa lifetime 2
   veos(config-ipsec-sa)#exit

   veos(config-ipsec)#sa policy sa-default
   veos(config-ipsec-sa)#exit

5. Bind or associate the IKE and SA policies together using a IPsec profile. Provide a shared-key, which must be common on both peers. The default profile assigns default values for all parameters that are not explicitly configured in the other profiles.
   **Example:** In this example, tunnel mode is set to transport. The IKE Policy **ike-peerRtr** and SA Policy **sa-peerRtr** are applied to profile **peer-Rtr**. Dead Peer Detection is enabled and configured to delete the connection when the peer is down for more than 50 seconds. The peer (**peer-Rtr**) is set to be the responder.

   veos(config-ipsec)#profile default
   veos(config-ipsec-profile)#ike-policy ikedefault
   veos(config-ipsec-profile)#sa-policy sadefault
   veos(config-ipsec-profile)#shared-key arista
   veos(config-ipsec)#profile peer-Rtr
   veos(config-ipsec-profile)#ike-policy ike-peerRtr
   veos(config-ipsec-profile)#sa-policy sa-peerRtr
   veos(config-ipsec-profile)#dpd 10 50 clear
   veos(config-ipsec-profile)#connection add
   veos(config-ipsec-profile)#mode transport

6. Configure the WAN interface to be the underlying interface for the tunnel. Specify an L3 address for the tunnel. If the L3 address is not specified, the vEOS Router cannot route packets using the tunnel.
Example:

```
veos(config)#interface Et1
veos(config-if-Et1)#no switchport
veos(config-if-Et1)#ip address 1.0.0.1/24
veos(config-if-Et1)#mtu 1500
```

7. Apply the IPsec profile to a new tunnel interface. Create the new tunnel interface as part of this step. **Example:** In this example, the new tunnel interface is *Tunnel0*. The new tunnel interface is configured to use IPsec, and the tunnel mode is set to GRE. Configure the other end of the tunnel also as a GRE-over-IPsec tunnel.

```
veos(config)#interface tunnel0
veos(config-if-Tu0)#ip address 1.0.3.1/24
veos(config-if-Tu0)#tunnel mode gre
veos(config-if-Tu0)#mtu 1400
veos(config-if-Tu0)#tunnel source 1.0.0.1
veos(config-if-Tu0)#tunnel destination 1.0.0.2
veos(config-if-Tu0)#tunnel ipsec profile vrouter
```

8. Create the GRE-over-IPsec tunnel interface in a VRF using the `vrf forwarding` command. Create the VRF, if needed, then create and configure the GRE tunnel interface. Make sure to specify the tunnel key that is unique across all tunnels.

   **Note:** If tunnels in different VRFs need to share the IPsec connection, specify the same **source**, **destination**, and **ipsec profile**.

```
veos(config)#vrf definition red
veos(config-vrf-red)#rd 1:3
veos(config-vrf-red)#interface tunnel0
veos(config-if-Tu0)#ip address 1.0.3.1/24
veos(config-if-Tu0)#vrf forwarding red
veos(config-if-Tu0)#tunnel mode gre
veos(config-if-Tu0)#mtu 1400
veos(config-if-Tu0)#tunnel source 1.0.0.1
veos(config-if-Tu0)#tunnel destination 1.0.0.2
veos(config-if-Tu0)#tunnel key 100
veos(config-if-Tu0)#tunnel ipsec profile vrouter
```

```
veos(config)#vrf definition blue
veos(config-vrf-blue)#rd 1:4
veos(config-vrf-blue)#interface tunnel1
veos(config-if-Tu1)#ip address 1.0.4.1/24
veos(config-if-Tu1)#vrf forwarding blue
veos(config-if-Tu1)#tunnel mode gre
veos(config-if-Tu1)#mtu 1400
veos(config-if-Tu1)#tunnel source 1.0.0.1
veos(config-if-Tu1)#tunnel destination 1.0.0.2
veos(config-if-Tu1)#tunnel key 200
veos(config-if-Tu1)#tunnel ipsec profile vrouter
```

9. Configure the GRE-over-IPsec tunnel on the peer router.

**Configuring VTI IPsec Tunnels**

The vEOS Router gives the ability to configure VTI IPsec tunnels between a vEOS Router instance and a third party peer router instance (such as a Palo Alto firewall VM). First, complete the set up of the tunnel on the vEOS Router instance, then set up the other end of the tunnel on the third party peer router instance.
Palo Alto Firewall VM Configuration

Use this configuration when pairing a Palo Alto firewall VM instance and vEOS Router instance as tunnel endpoints of an IPsec VTI IPsec tunnel.

**Note:** Refer to the Palo Alto firewall VM documentation for configuration details, including the different interfaces to use to complete the configuration and all the parameters and options.

Supported Tunnel Types

Set up IPsec VTI tunnels when using the Palo Alto firewall VM as a peer router instance with a vEOS Router instance. IPsec GRE-over-IPsec tunnels using this combination of router instances as peers is not permitted.

Configuration Guidelines

The following are guidelines to follow when configuring the Palo Alto firewall VM.

- **IP address settings.**
  
  Configure the first interface to be configured (typically named `eth0`), as the management interface. Use the public IP address on this interface to open the GUI of the Palo Alto firewall VM.

- **Management interface.**
  
  Use this interface only for control plane traffic.

- **Management profile.**
  
  When configuring the profile, select all of the protocols allowed on the management interface.

Procedure

1. Create a new management profile. Select all of the protocols allowed on the management interface.

2. Create a new tunnel interface and specify the following parameters.

   - **Name:** (for example, `tunnel 1`.)
   - **Virtual router:** (Select the existing virtual router.)
   - **Security Zone:** (Select the layer 3 internal zone, which is the zone from which the traffic originates.)
   - **IP address:** (Tunnel IP address.)

3. Add a new IKE Crypto profile and specify the IKE options.

   **Note:** Make sure the settings match the IKE settings on the other end of the tunnel (the vEOS Router instance). This setting ensures that the IKE negotiation is successful.

   - **Name:** (can be any name.)
   - **Virtual router:** (Select the existing virtual router.)
   - **Security Zone:** (Select the layer 3 internal zone, which is the zone from which the traffic originates.)
   - **IP address:** (Tunnel IP address.)

4. Configure the IKE gateway.

   **Note:** Make sure the pre-shared key matches the key defined on the other end of the tunnel (the vEOS Router instance).

5. Add a new IKE Crypto profile for the IKE options.

   **Note:** Make sure the settings match the IKE settings on the other end of the tunnel (the vEOS Router instance). This setting ensures that the IKE negotiation of IPsec SAs is successful.
6. Create a new IPsec tunnel, and select the tunnel interface, IKE gateway, IKE crypto profile, and IKE crypto profile defined earlier in the procedure. Selecting these elements binds them to the new tunnel interface.

   ✍️ Note: Enter the destination IP address of the tunnel interface of the vEOS Router in the Destination IP option (one of the Tunnel Monitor settings on the Palo Alto firewall VM).

7. Create a new static route for the network that is behind the remote tunnel endpoint. This new static route ensures that the traffic flows through the tunnel to the other tunnel endpoint.

8. **Commit** (save) the configuration.

**vEOS and Palo Alto Firewall VM Pairing (VTI IPsec Tunnel)**

The following example shows a VTI IPsec tunnel between a vEOS Router instance and a third party Palo Alto firewall VM router instance.

**Running Configuration for vEOS1**

```
ip security
   ike policy ikebranch1
      integrity sha256
dh-group 15
!
sa policy sabranch1
   sa lifetime 2
   pfs dh-group 14
!
profile hq
   ike-policy ikebranch1
   sa-policy sabranch1
   connection add
   shared-key keyAristaHq
   dpd 10 50 clear
!
interface Ethernet1
   no switchport
   ip address 1.0.0.1/24
!
interface Management1
   ip address dhcp
!
interface Tunnel1
   mtu 1404
   ip address 1.0.3.1/24
tunnel mode ipsec
tunnel source 1.0.0.1
tunnel destination 1.0.0.2
tunnel ipsec profile hq
!
```

**Running Configuration on Palo Alto Firewall VM**

```
"ike": {
   "crypto-profiles": {
      "ike-crypto-profiles": [
      
      
      
      
      "@name": "veos12-IKE-Phase1",
      "hash": {"member": "sha512"}

```
vEOS Router Configuration

Use this procedure to configure VTI IPsec tunnels on an Arista router instance. Complete the procedure, then configure the other tunnel endpoint on the third party peer router.

Note: The vEOS Router by default uses IKE version 2 for all IPsec tunnels. To configure a VTI IPsec tunnel that uses IKE version 1, explicitly configure the vEOS Router instance to use IKE version 1.

Procedure

Complete the following steps to configure a vEOS Router instance to share a VTI IPsec tunnel.

To use IKE version 1, complete the section below, then continue with the steps below. To use IKE version 2, which is the default version, start with Step 1 below.

veos(config)#ip security
veos(config-ipsec)#ike policy ike-peerRtr
veos(config-ipsec-ike)#version 1

1. Use this command to enter IP security mode.

veos(config)#ip security

2. Create an IKE Policy to communicate with the peer to establish IKE Phase 1 options. There is the option of configuring multiple IKE policies.

The default IKE Policy values are:

• Encryption - AES256
• Integrity - SHA256
• DH group - Group 14
• IKE lifetime - 8 hours

Example:

veos(config)#ip security
veos(config-ipsec)#ike policy ike-vrouter-PA
veos(config-ipsec-ike)#integrity sha512
veos(config-ipsec-ike)#encryption aes256
veos(config-ipsec-ike)#dh-group 20

3. If the router is behind a NAT, configure the local-id with the local public IP address.
4. Create an IPsec Security Association policy in the data path for encryption and integrity. There is the option of enabling Perfect Forward Secrecy by configuring a DH group to the SA.
   **Example:** In this example, AES256 is used for encryption, SHA 256 is used for integrity, and Perfect Forward Secrecy is enabled (the DH group is 20).

   ```
   veos(config-ipsec-ike)#local-id <public ip address>
   veos(config-ipsec)#sa policy sa-vrouter-PA
   veos(config-ipsec)#esp encryption aes256
   veos(config-ipsec)#esp integrity sha256
   veos(config-ipsec)#sa lifetime 2
   veos(config-ipsec)#pfs dh-group 20
   ```

5. Bind or associate the IKE and SA policies together using an IPsec profile. Provide a shared-key, which must be common on both peers. The default profile assigns default values for all parameters that are not explicitly configured in the other profiles.
   **Example:** In this example, the IKE Policy *ike-vrouter-PA* and SA Policy *sa-vrouter-PA* are applied to profile *vrouter-PA*. Dead Peer Detection is enabled and configured to delete the connection when the peer is down for more than 30 seconds.

   ```
   veos(config-ipsec)#profile vrouter-PA
   veos(config-ipsec-profile)#ike-policy ike-vrouter-PA
   veos(config-ipsec-profile)#sa-policy sa-vrouter-PA
   veos(config-ipsec-profile)#connection start
   veos(config-ipsec-profile)#shared-key Arista1234
   veos(config-ipsec-profile)#dpd 10 30 clear
   ```

6. Create a tunnel interface for the VTI tunnel. When tunnel mode is set to IPsec, configure a tunnel key on the vEOS Router instance to ensure that traffic can be forwarded through the tunnel.
   **Example:**

   ```
   veos(config)#interface Tunnel1
   veos(config-if-Tu1)#mtu 1400
   veos(config-if-Tu1)#ip address 1.0.3.1/24
   veos(config-if-Tu1)#tunnel mode ipsec
   veos(config-if-Tu1)#tunnel source 10.2.201.149
   veos(config-if-Tu1)#tunnel destination 10.3.31.30
   veos(config-if-Tu1)#tunnel ipsec profile vrouter-PA
   ```

   Configure the VTI IPsec tunnel on the peer router (see *Palo Alto Firewall VM Configuration* on page 107).

**vEOS Router Show Commands**

The vEOS Router has show commands to view IPsec connections and IPsec profiles on vEOS Router instances.

**View all Existing IPsec Connections**

Use the `veos#show ip security connection` command to view all existing IPsec connections.

**Example**

```
veos#show ip security connection
Tunnel   Source  Dest    Status   Uptime
Tunnel10 1.0.0.1  1.0.0.2 Established 14 minutes

Input    Output    Reauth Time
589 bytes 608 bytes 8 hours
7 pkts    36 pkts
```
veos#show ip security connection detail
source address 1.0.0.1, dest address 1.0.0.2
    Initiator SPI 1731161283:
        request id 1, mode transport replay-window 32, seq 0x0
        stats errors:
            replay-window 0, replay 0, integrity_failed 0
        lifetime config:
            softlimit 18446744073709551615 bytes, hardlimit 18446744073709551615 bytes
            softlimit 18446744073709551615 pkts, hardlimit 18446744073709551615 pkts
            expire add 0 secs, hard 0 secs
        lifetime current:
            589 bytes, 7 pkts

    Responder SPI 2180490447:
        request id 1, mode transport replay-window 32, seq 0x0
        stats errors:
            replay-window 0, replay 0, integrity_failed 0
        lifetime config:
            softlimit 18446744073709551615 bytes, hardlimit 18446744073709551615 bytes
            softlimit 18446744073709551615 pkts, hardlimit 18446744073709551615 pkts
            expire add 0 secs, hard 0 secs
        lifetime current:
            608 bytes, 7 pkts

View Profiles Currently Being Used by IPsec Connections

Use the `show ip sec applied-profile` command to view the IPsec profiles that have been applied to existing IPsec connections.

veos#show ip sec applied-profile
Profile Name   Interface
Arista       Tunnel0

IPsec Show Commands

The vEOS Router provides commands to view all current or established IPsec tunnels and to view all profiles currently in use by established tunnels.

The show commands are:

* `show ip security connection`
* `show ip security connection detail`

Examples

The example below shows the use of the `show ip security connection` command to view a summary of all current (established) IPsec tunnels.

veos#show ip security connection
Tunnel  Source   Dest   Status    Uptime
Tunnel0  1.0.0.1  1.0.0.2 Established 14 minutes

Input       Output       Reauth Time
The example below shows the use of the `show ip security connection detail` command to view the details for a specified IPsec tunnel.

```
veos#show ip security connection detail
source address 1.0.0.1, dest address 1.0.0.2
  Inbound SPI 0x672F6CC3:
    request id 1, mode transport replay-window 32, seq 0x0
    stats errors:
      replay-window 0, replay 0, integrity_failed 0
    lifetime config:
      softlimit 18446744073709551615 bytes, hardlimit 18446744073709551615
      pkts
      expire add 0 secs, hard 0 secs
    lifetime current:
      589 bytes, 7 pkts
  Outbound SPI 0xc5f3c373:
    request id 1, mode transport replay-window 32, seq 0x0
    stats errors:
      replay-window 0, replay 0, integrity_failed 0
    lifetime config:
      softlimit 18446744073709551615 bytes, hardlimit 18446744073709551615
      pkts
      expire add 0 secs, hard 0 secs
    lifetime current:
      608 bytes, 7 pkts
```

The example below shows the use of the `show ip sec applied-profile` command to view all profiles currently in use by established tunnels.

```
veos#show ip sec applied-profile
Profile Name | Interface
-------------|-----------
Arista       | Tunnel10
```

**vEOS Routers and CSR**

Use this configuration process to set up GRE-over-IPsec tunnels on CSR peer routers. Procedures are provided for configuration using IKE version 1, or IKE version 2. Make sure to use the correct procedure based on the selected version of IKE.

**CSR Configuration**

The configuration of VTI IPsec tunnels on CSR peer router instances is almost identical to the configuration of GRE-over-IPsec tunnels on CSR peer router instances. The only difference in the configurations is tunnel mode.

For VTI IPsec tunnels, tunnel mode must be set to `ipsec` instead of `gre` (for GRE-over-IPsec tunnels, tunnel mode must be set to `gre`.)
This example shows a basic VTI IPsec tunnel configuration for a CSR peer router instance.

**Example**

```
CSR(config)#interface Tunnel0
CSR(config-if)#ip address 1.0.3.1 255.255.255.0
CSR(config-if)#tunnel source 10.3.31.30
CSR(config-if)#tunnel destination 10.2.201.149
CSR(config-if)#tunnel mode ipsec ipv4
CSR(config-if)#tunnel protection ipsec profile vrouter-ikev1-ipsec-profile
```

*Note:* Make sure you use the correct procedure based on the version of IKE you need to use.

**Sharing IPsec Connections**

On CSR, the user can configure multiple GRE tunnels to use the same IPsec connection.

The user needs to add an extra shared keyword after the profile name on every tunnel interface that is to be shared.

```
veos(config)#interface Tunnel0
veos(config-if)#tunnel protection ipsec profile vrouter-ikev2-ipsec-profile
   shared
veos(config-if)#exit
```

**IKEv1 Configuration**

The CSR configuration to create a GRE over IPsec tunnel is similar the vEOS Router setup using *ikev1 version*.

To ensure that the v EOS Router can establish a tunnel with CSR, it needs to set the ikev1 version as follows:

```
veos(config)#ip security
veos(config-ipsec)#ike policy ike-peerRtr
veos(config-ipsec-ike)#version 1
```

1. Enter the configuration terminal mode to configure IPsec.
   
   CSR#config terminal

2. Configure a pre-shared key for the vEOS Router and CSR to authenticate each other. Create a keyring to hold the keys.
   
   ```
   CSR(config)#crypto keyring vrouter-keyring
   CSR(conf-keyring)#pre-shared-key address 1.0.0.2 key arista
   ```

3. Create an ISAKMP policy. The policy’s function is to communicate with the peer to establish IKE Phase 1. In the example below, a policy with AES256 is created with the following parameters: SHA1, DH group 15, authentication pre-share, and a lifetime of 28800 seconds.
   
```
CSR(config)#crypto isakmp policy 1
CSR(config-isakmp)#encr aes 256
CSR(config-isakmp)#hash sha
CSR(config-isakmp)#authentication pre-share
CSR(config-isakmp)#group 15
CSR(config-isakmp)#lifetime 28800
```

4. Create an ISAKMP profile associated with the vEOS Router to match its outside IP Address and the keyring that was created earlier to identify the pre-shared secret.
   
   ```
   CSR(config)#crypto isakmp profile vrouter-ikev1-isakmp-profile
   CSR(config-isa-prof)#keyring vrouter-keyring
   ```
5. Create the IPsec transform-set configuration settings. The transform-set defines the encryption and hash algorithm for the child/IPsec SA. This example creates a transform-set with AES cipher for the ESP encryption and SHA1 for the authentication. The mode for the IPsec is set to `transport` mode.

   ```
   CSR(config)#crypto ipsec transform-set vrouter-tset esp-aes 256 esp-sha-hmac
   CSR(cfg-crypto-trans)#mode transport
   ```

6. Create the IPsec profile which includes the `transform-set`, `SA idle time`, `lifetime`, and replay windows used to create the child SA.

   ```
   CSR(config)#crypto ipsec profile vrouter-ikev1-ipsec-profile
   CSR(ipsec-profile)#set security-association idle-time 3600
   CSR(ipsec-profile)#set security-association dummy seconds 3600
   CSR(ipsec-profile)#set transform-set vrouter-tset
   CSR(ipsec-profile)#set isakmp-profile vroute-ikev1-isakmp-profile
   ```

7. Configure the WAN interface as the underlying interface for the tunnel. To be able to route packets, the tunnel is given an L3 IP address.

   ```
   CSR(config)#interface GigabitEthernet2
   CSR(config-if)#ip address 1.0.0.2 255.255.255.0
   CSR(config-if)#mtu 9001
   CSR(config-if)#negotiation auto
   ```

8. Apply the IPsec profile to a tunnel interface. The example creates a tunnel interface (`Tunnel0`) and configures the tunnel interface to use IPsec.

   ```
   CSR(config-if)#exit
   CSR(config)#interface Tunnel0
   CSR(config-if)#ip address 1.0.3.1 255.255.255.0
   CSR(config-if)#tunnel source 1.0.0.2
   CSR(config-if)#tunnel destination 1.0.0.1
   CSR(config-if)#tunnel protection ipsec profile vrouter-ikev1-ipsec-profile
   CSR(config-if)#exit
   ```

### IKEv2 Configuration

The CSR configuration to create a GRE over IPsec tunnel is similar to the vEOS Router setup using `ikev2 version`.

By default, the vEOS Router is configured to run in IKEv2 version. Make sure the version is not set to 1 under the `ike` policy. The configuration steps for CSR IKEv2 are a bit different to that of IKEv1.

Complete the following steps to configure the CSR.

1. Enter the configuration terminal mode to configure IPsec.

   ```
   CSR#configure terminal
   ```

2. Create a pre-shared key for CSR and the vEOS Router to authenticate each other. Create a keyring to hold the keys. Specify the peer vEOS Router under which the keys and matching IP address of peer are configured.

   ```
   CSR(config)#crypto keyring vrouter-ikev2-keyring
   CSR(config-keyring)#pre-shared-key address 1.0.0.2 key arista
   ```
3. Create an IKEv2 proposal to specify the encryption, integrity, and group. In the example, it specifies AES256, SHA1, and DH group 14.

```plaintext
CSR(config)#crypto ikev2 proposal vrouter-ikev2-proposal
CSR(config-ikev2-proposal)#encryption aes-cbc-256
CSR(config-ikev2-proposal)#integrity sha1
CSR(config-ikev2-proposal)#group 14
CSR(config-ikev2-proposal)#exit
```

4. Create an IKEv2 policy and attach the proposal created in the previous step.

```plaintext
CSR(config)#crypto ikev2 policy vrouter-ikev2-policy
CSR(config-ikev2-policy)#match fvrf any
CSR(config-ikev2-policy)#proposal vrouter-ikev2-proposal
CSR(config-ikev2-policy)#exit
```

5. Create an IKEv2 profile and specify the match identity for the remote peer's address, authentication pre-share, and the keyring that was previously created.

```plaintext
CSR(config)#crypto ikev2 profile vrouter-ikev2-profile
CSR(config-ikev2-profile)#match fvrf any
CSR(config-ikev2-profile)#match identity remote address 1.0.0.1 255.255.255.255
CSR(config-ikev2-profile)#authentication remote pre-share key arista
CSR(config-ikev2-profile)#authentication local pre-share key arista
CSR(config-ikev2-profile)#exit
```

6. Create the IPsec transform-set configuration settings. This step is similar to the step in IKEv1 configuration. The transform-set defines the encryption and hash algorithm for the child/IPsec SA. The example creates a transform-set with AES cipher for the ESP encryption and SHA1 for the authentication. The mode for the IPsec is set to the transport mode.

```plaintext
CSR(config)#crypto ipsec transform-set vrouter-tset esp-aes 256 esp-sha-hmac
CSR(cfg-crypto-trans)#mode transport
```

7. Create the IPsec profile similar to IKEv1. This profile includes the transform-set, SA idle time, lifetime, and replay windows that are used to create the child SA and specifies the IKEv2 profile to use.

```plaintext
CSR(config)#crypto ipsec profile vrouter-ikev2-ipsec-profile
CSR(ipsec-profile)#set security-association idle-time 3600
CSR(ipsec-profile)#set security-association dummy seconds 3600
CSR(ipsec-profile)#set transform-set vrouter-tset
CSR(ipsec-profile)#set ikev2-profile vrouter-ikev2-profile
CSR(ipsec-profile)#exit
```

8. Configure the interface to use as the underlying interface for the tunnel. To be able to route packets, the tunnel is given an L3 IP address.

```plaintext
CSR(config)#interface GigabitEthernet2
CSR(config-if)#ip address 1.0.0.1 255.255.255.0
CSR(config-if)#negotiation auto
```

9. Apply the IPsec profile to a tunnel interface. The example creates a tunnel interface (Tunnel0) and configures the tunnel interface to use IPsec.

```plaintext
CSR(config-if)#exit
CSR(config)#interface Tunnel0
CSR(config-if)#ip address 1.0.3.1 255.255.255.0
CSR(config-if)#tunnel path-mtu-discovery
CSR(config-if)#tunnel source 1.0.0.1
CSR(config-if)#tunnel destination 1.0.0.2
```
vEOS Router (GRE-over-IPsec Tunnel)
The IPsec tunnels represented in these examples include GRE-over-IPsec tunnels on vEOS Router instances.

**Running Configuration for vEOS**
```
ip security
ike policy ikebranch1 encryption aes256 dh-group 15
!
sa policy sabranch1 sa lifetime 2
pfs dh-group 14
!
profile hq
ike-policy ikebranch1 sa-policy sabranch1 connection add
shared-key keyAristaHq dpd 10 50 clear
!
interface Tunnel1
ip address 1.0.3.1/24 tunnel mode gre tunnel source 1.0.0.1
tunnel destination 1.0.0.2 tunnel ipsec profile hq

interface Ethernet1 no switchport
ip address 1.0.0.1/24
```

vEOS Router (VTI IPsec Tunnel)
The IPsec tunnels represented in these examples include VTI IPsec tunnels between vEOS Router instances and third party CSR router instances.

**Running Configuration for vEOS**
```
ip security
ike policy ikebranch1
   encryption aes256
da dh-group 15
!
sa policy sabranch1
   sa lifetime 2
   pfs dh-group 14
!
profile hq
ike-policy ikebranch1
   sa-policy sabranch1
   connection add
   shared-key keyAristaHq
dpd 10 50 clear
!
interface Tunnel1
ip address 1.0.3.1/24
tunnel mode ipsec
tunnel source 1.0.0.1
tunnel destination 1.0.0.2
tunnel key 100
tunnel ipsec profile hq
```
CSR Commands

The CSR router has show commands for several IPsec tunnel elements on CSR router instances.

CSR Router Show Commands

Describes the available CSR Router show commands and their example outputs.

Use the different show commands for CSR router instances to do the following:

- View all Existing ISAKMP SAs
- View all Existing IPsec SAs
- View Crypto (Encryption) Session Details
- View IKEv2 SAs
- View IKEv2 SA Details

View all Existing ISAKMP SAs

Use the `show crypto isakmp sa` command to view the ISAKMP SAs for all existing or current IPsec connections.

Example

```
CSR#show crypto isakmp sa
IPv4 Crypto ISAKMP SA
  dst src state conn-id status
  1.0.0.1 1.0.0.2 QM_IDLE 1331 ACTIVE
  vrouter-ikev1-isakmp-profile
IPv6 Crypto ISAKMP SA
```

View all Existing IPsec SAs

Use the `show crypto ipsec sa` command to view the IPsec SAs for all existing or current IPsec connections.

Example

```
CSR#show crypto ipsec sa
interface: Tunnel0
  Crypto map tag: Tunnel0-head-0, local addr 1.0.0.2
protected vrf: (none)

  local ident (addr/mask/prot/port):
  (1.0.0.2/255.255.255.255/47/0)
  remote ident (addr/mask/prot/port):
  (1.0.0.1/255.255.255.255/47/0)
  current_peer 1.0.0.1 port 500
  PERMIT, flags={origin_is_acl,}
  #pkts encaps: 1, #pkts encrypt: 1, #pkts digest: 1f
  #pkts decaps: 1, #pkts decrypt: 1, #pkts verify: 1
  #pkts compressed: 0, #pkts decompressed: 0
  #pkts not compressed: 0, #pkts compr. failed: 0
  #pkts not decompressed: 0, #pkts decompress failed: 0
```
View Crypto (Encryption) Session Details

Use the `show crypto session detail` command to view details about the crypto session for all current IPsec connections.

**Example**

```
CSR#show crypto session detail
```

Crypto session current status

Code:  C - IKE Configuration mode, D - Dead Peer Detection
      K - Keepalives, N - NAT-traversal, T - cTCP encapsulation
      X - IKE Extended Authentication, F - IKE Fragmentation
      R - IKE Auto Reconnect

Interface: Tunnel0
Profile: vrouter-ikev1-isakmp-profile
Uptime: 00:20:23
Session status: UP-ACTIVE
Peer: 1.0.0.1 port 500 fvrf: (none) ivrf: (none)
Phase 1

---

**Session ID:** 0

**IKEv1 SA:**
- **local 1.0.0.2/500**
- **remote 1.0.0.1/500**
- **Active**
- **Capabilities:** (none)
- **connid:** 1332
- **lifetime:** 07:39:35
- **IPSEC FLOW:** permit 47 host 1.0.0.2 host 1.0.0.1
- **Active SAs:** 2, **origin:** crypto map
- **inbound:** #pkts dec'ed 42 drop 0 life (KB/Sec)
- **Outbound:** #pkts enc'ed 44 drop 0 life (KB/Sec)

---

**View IKEv2 SAs**

Use the `show crypto ikev2 sa` command to view summary information about all IKE version 2 SAs in use by existing IPsec connections.

**Example**

```
CSR#show crypto ikev2 sa
IPv4 Crypto IKEv2 SA

Tunnel-id  Local Remote fvrf/ivrf Status
1 3.3.3.3/500 3.3.3.1/500 none/none READY

Encr: AES-CBC, keysize: 128, PRF: sha256, Hash: SHA96,
DH Grp:14, Auth sign: PSK, Auth verify: PSK
Life/Active Time: 86400/5349 sec
```

**IPv6 Crypto IKEv2 SA**

---

**View IKEv2 SA Details**

Use the `show crypto ikev2 sa detailed` command to view details about all IKE version 2 SAs in use by existing IPsec connections.

**Example**

```
CSR#show crypto ikev2 sa detailed
IPv4 Crypto IKEv2 SA

Tunnel-id  Local Remote fvrf/ivrf Status
1 3.3.3.3/500 3.3.3.1/500 none/none READY

Encr: AES-CBC, keysize: 128, PRF: sha256, Hash: SHA96,
DH Grp:14, Auth sign: PSK, Auth verify: PSK
Life/Active Time: 86400/5358 sec
CE id: 1351, Session-id: 6
Status Description: Negotiation done
Local spi: 9FA0B7B1F7746E69 Remote spi: 4B1652D32691E8AF
Local id: 3.3.3.3 Remote id: 3.3.3.1
Local req msg id: 4 Remote req msg id: 8
Local next msg id: 4 Remote next msg id: 8
Local req queued: 4 Remote req queued: 8
Local window: 5 Remote window: 1
DPD configured for 0 seconds, retry 0
Fragmentation not configured.
Extended Authentication not configured.
NAT-T is not detected
Cisco Trust Security SGT is disabled
```
vEOS Routers and AWS Specific Cloud Configuration

Describes the configuration steps for an AWS specific cloud on a vEOS Router instance.

IPsec Between the vEOS Router and AWS Specific Cloud Configuration

Describes the steps and the running configuration for setting up an IPsec connection between the vEOS Router and the AWS Specific Cloud. The AWS Specific Cloud only supports IKE1 and not IKE2.

The following configurations are for the minimum requirement of AES128, SHA1, and DH Group 2. These can be modified to take advantage of AES256, SHA256, or other DH groups such as 5, 14-17, and 24.

Running-configuration of the vEOS Router and AWS Specific Cloud

The sample configuration below sets up the running configuration of the vEOS Router and AWS Specific Cloud. In the configuration, the local-id is the external IP of the router when it is behind a NAT device, and the tunnel destination is the external IP of the AWS Specific Cloud.

```
ip security
  ike policy AWS-IKE1
    integrity sha1
    version 1
    local-id 52.165.228.195
  !
  ike policy ikedefault
    encryption aes256
  !
  sa policy AWS-SA1
    esp encryption aes128
    esp integrity sha1
    pfs dh-group 14
  !
profile AWS-profile
  ike-policy AWS-IKE1
  sa-policy AWS-SA1
  connection start
  sharded-key LwYbARmDJmpFGAOrAbPGk2uQiWwvbmfU
  !
profile default
  ike-policy
  sa-policy AWS-SA1
  shared-key arista
  !
interface Tunnel1
  ip address 169.254.11.162/30
  tunnel mode ipsec
  tunnel source 10.2.0.4
  tunnel destination 52.53.75.160
  tunnel ipsec profile AWS-profile
```
AWS Specific Cloud Configuration

1. Internet Key Exchange Configuration

The address of the external interface for a customer gateway must be a static address. The customer gateway can reside behind a device performing network Address Translation (NAT). To ensure that NAT Transversal (NAT-T) can function, add and update the firewall rules, allow UDP port 4500. Disable NAT-T if the customer gateway is not behind a NAT gateway.

- **Authentication Method:** Pre-Shared Key
- **Pre-Shared Key:** LwYbARmDJmpFGOrAbPGk2uQiWwvbmfU
- **Authentication Algorithm:** sha1
- **Encryption Algorithm:** aes-128-cbc
- **Lifetime:** 28800 seconds
- **Phase 1 Negotiation Method:** main
- **Perfect Forward Secrecy:** Diffie-Hellman Group 2

AWS Specific Cloud Configuration Modifications

1. Internet Key Exchange SA Configuration

The address of the external interface for the customer gateway must be a static address. The customer gateway can reside behind a device performing Network Address Translation (NAT). To make sure that NAT traversal (NAT-T) functions correctly, add or update the firewall rule to allow UDP port 4500. Disable NAT-T if the customer gateway is not behind a NAT gateway.

Use the following sample configuration files to set up an Internet key exchange SA configuration.

- **Authentication Method:** Pre-shared Key
- **Pre-shared Key:** LwYbARmDJmpFGAOArAbPGk2uQiWwvbmfU
- **Authentication Algorithm:** sha1
- **Encryption Algorithm:** aes-128-cbc
- **Lifetime:** 28800 seconds
- **Phase 1 Negotiation Mode:** main
- **Perfect Forward Secrecy:** Diffie-Hellman Group 2

2. IPsec Configuration

Use the following sample configuration files to configure the IPsec. Modification of the sample configuration files may be need to take advantage of additionally supported IPsec parameters for encryption, such as AES256 and other DH groups like 2, 5, 14-18, 22, 23, and 24.

- **Protocol:** esp
- **Authentication Algorithm:** hmac-sha-96
- **Encryption Algorithm:** aes-128-cbc
- **Lifetime:** 3600 seconds
- **Mode:** tunnel
- **Perfect Forward Secrecy:** Diffie-Hellman Group 2

The IPsec Dead Peer Detection (DPD) is enabled on the AWS Specific Cloud endpoint. Configure the DPD on your endpoint as follows:

- **DPD interval:** 10
- **DPD Retries:** 3
The IPsec Encapsulating Security Payload (ESP) inserts additional headers to transmit the packets. These headers require additional space, which reduces the amount of space available to transmit application data. The following configuration is recommended on the customer gateway to limit the impact of this behavior:

- **TCP MSS Adjustment**: 1379 bytes
- **Clear Don’t fragment Bit**: enabled
- **Fragmentation**: Before encryption

3. **Tunnel Interface Configuration**

Configure the customer gateway with a tunnel interface that associates with the IPsec tunnel. All traffic transmitted to the tunnel interface is encrypted and transmitted to the virtual private gateway.

The customer gateway and the virtual private gateway each have two addresses that relate to this IPsec tunnel. Each one contains an outside address, where the encrypted traffic is exchanged. Both gateways also contain an inside address associated with the tunnel interface. The customer gateway outside IP address is provided upon creation of the customer gateway. To change the IP address of the customer gateway, create a new customer gateway. The customer gateway inside IP address must be configured on the interface tunnel.

**Outside IP Addresses**

- **Customer Gateway**: 52.165.228.195
- **Virtual Private Gateway**: 52.53.75.160

The customer gateway IP address is the IP address of the firewall that the vEOS instance in the DC with NAT behind.

The virtual private gateway IP address is the external IP address of the AWS Specific Cloud.

**Inside IP Addresses**

- **Customer Gateway**: 169.254.11.162/30
- **Virtual Private Gateway**: 169.254.11.161/30

The virtual private gateway IP address is the tunnel IP address of the AWS Specific Cloud.

4. **Static Routing Configuration**

The router traffic between the internal network and the VPC an AWS Specific Cloud, add a static router to the vEOS Router.

**Next Hop**: 169.254.11.162

Any subnet that requires a route to DC must have a route pointing to the AWS Specific Cloud tunnel IP address.

For traffic destined to the Internet Network, add static routes on the VGW.
ECMP

Equal-cost multi-path routing (ECMP) is a routing plan of action where next-hop packet forwarding to a single destination takes place over multiple "best paths" which tie for top place in routing metric calculations. Use multi-path routing in conjunction with most routing protocols, because it is a per-hop decision limited to a single router.

Adding ECMP

Describes how to add ECMP to new or existing VMs.

1. Have the following line in a device's running-configuration.

   If an instance is created with an older, pre-vEOS 4.20.5 image, add the command line in the example below.

   If an instance is created with vEOS 4.20.5 or later image, there is no need for additional configuration changes because the command line appears in the configuration by default.

   ```
   agent KernelFib environment KERNELFIB_PROGRAM_ALL_ECMP='true'
   ```

2. Reload the device or restart the KernelFib agent via `agent KernelFib terminate`.
   This step is needed only if the instance was created with an older, pre-vEOS 4.20.5 image.

3. To enable ECMP in a routing protocol, issue the `maximum-paths <#>` command inside the routing protocol used.

   ```
   veos#configure terminal
   veos(config)#router bgp 65112
   veos(config-router-bgp)#maximum-paths 16
   ```

4. When ECMP starts, and there are multiple routes, display output may be similar to the following example.

   ```
   veos#show ip route 10.4.3.0
   ```

<table>
<thead>
<tr>
<th>VRF: default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Codes: C - connected, S - static, K - kernel,</td>
</tr>
<tr>
<td>O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1,</td>
</tr>
<tr>
<td>E2 - OSPF external type 2, N1 - OSPF NSSA external type 1,</td>
</tr>
<tr>
<td>N2 - OSPF NSSA external type 2, B I - iBGP, B E - eBGP,</td>
</tr>
<tr>
<td>R - RIP, I L1 - IS-IS level 1, I L2 - IS-IS level 2,</td>
</tr>
<tr>
<td>O3 - OSPFv3, A B - BGP Aggregate, A O - OSPF Summary,</td>
</tr>
<tr>
<td>NG - Nexthop Group Static Route, V - VXLAN Control Service,</td>
</tr>
<tr>
<td>DH - DHCP client installed default route, M - Martian</td>
</tr>
</tbody>
</table>

   | S 10.4.3.0/24 [1/0] via 190.19.11.2, Tunnel1 |
   | via 190.19.11.102, Tunnel3 |
5. To determine the route that the interface traffic takes to specific addresses, issue the `bash ip route get <address>` command to determine which link the traffic uses. In the following example, traffic to **10.4.3.5** takes Tunnel1, while traffic to **10.4.3.6** takes Tunnel3.

```
veos#bash ip route get 10.4.3.5
10.4.3.5 via 190.19.11.2 dev tun1 src 190.19.11.1
    cache
veos#bash ip route get 10.4.3.6
10.4.3.6 via 190.19.11.102 dev tun3 src 190.19.11.101
    cache
```

For additional information regarding ECMP, refer to the current release notes.
Index

A
access-key-id (vEOS-AWS) 25
active-directory credential email subscription-id (vEOS-Azure) 26
Attaching the New Network Interfaces to Instances 55
azure (vEOS-Azure) 26

B
backup-gateway (vEOS - Azure) 27
bfd source-interface (vEOS) 27

C
Cloud Configuration 16
cloud high-availability shutdown 28
cloud high-availability vEOS) 28
cloud provider aws 29
cloud provider azure 29
Cloud Provider Side Helpful Tips 18
cloud proxy (vEOS) 30
Configuring BFD 19
Configuring Cloud High Availability 18
Configuring the AWS CloudWatch Logs Agent 52
Configuring the Cloud Provider 16
Configuring the Cloud Proxy 16
Configuring the Recovery Time 19
Configuring the Route Table of the AWS Router 56

E
Examples of Running-configuration for GRE-over-IPsec Tunnels 100

H
http (vEOS) 30
https (vEOS) 31

I
IPsec Show Commands 112

J
JSON-Based Cloud High Availability Configurations and Equivalent CLI 20

L
Launching vEOS Router Instances Using AWS CloudFormation 40
Launching vEOS Router Instances Using EC2 AWS Marketplace 44

P
peer (vEOS) 32
primary gateway (vEOS - Azure) 33
proxy 33

R
recovery (cloud HA peer) 31
recovery wait-time (vEOS) 34
region (vEOS-AWS) 34
Requirements when Behind a NAT 96

S
secret-access_key (vEOS-AWS) 35
show cloud high-availability (vEOS) 36
show cloud high-availability routes 36
show cloud provider aws (vEOS-AWS) 37
show cloud provider azure (vEOS-Azure) 38
show cloud proxy (vEOS) 38
Supported Tunnel Types 96

U
Using IPsec on vEOS and Third Party Devices 103
Using IPsec on vEOS Router Instances 97

V
vEOS and Palo Alto Firewall VM 108
vEOS Router and Palo Alto Firewall VM 104
vEOS Router Configuration 104
vEOS Routers and AWS Specific Cloud Configuration 121
vEOS Routers and CSR 113