# Aviatrix Provider

The Aviatrix provider is used to interact with Aviatrix organization resources.

This provider allows you to manage your Aviatrix organization's gateways, tunnels, and other resources easily. It needs to be configured with the proper credentials before it can be used.

Use the navigation to the left to read about the available resources.

**NOTE:** Although *version* is an optional argument, we highly recommend users to specify the proper Aviatrix provider release version corresponding to their Controller version in order to avoid potential compatibility issues. Please see the compatibility chart (https://www.terraform.io/docs/providers/aviatrix/guides/release-compatibility.html) for full details. For more information on versioning, a native Terraform provider argument, see here (https://www.terraform.io/docs/configuration/providers.html#version-provider-versions).

## Example Usage

```
# Configure Aviatrix provider
provider "aviatrix" {
  controller_ip           = "1.2.3.4"
  username                = "admin"
  password                = "password"
  skip_version_validation = false
  version                 = "2.5.0"
}

# Create an access account
resource "aviatrix_account" "myacc" {
  # ...
}
```

## Argument Reference

The following arguments are supported:

- `controller_ip` - (Required) Aviatrix controller's public IP.
- `username` - (Required) Aviatrix account username which will be used to login to Aviatrix controller.
- `password` - (Required) Aviatrix account password corresponding to above username.
- `skip_version_validation` - (Optional) Default: false. If set to true, it skips checking whether current Terraform provider supports current Controller version.
- `version` - (Optional) - Specify Aviatrix provider release version number. If not specified, Terraform will automatically pull and source the latest release.

# Import

Instances can be imported using the id, e.g.

```
$ terraform import aviatrix_instance.test myAviatrixInstanceID
```

# aviatrix_account

Use this data source to get the Aviatrix cloud account for use in other resources.

## Example Usage

```
# Aviatrix Account Data Source
data "aviatrix_account" "foo" {
  account_name = "username"
}
```

## Argument Reference

The following arguments are supported:

- `account_name` - (Required) Account name. This can be used for logging in to CloudN console or UserConnect controller.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `cloud_type` - Type of cloud service provider. (Only AWS is supported currently. Value of 1 for AWS.)
- `aws_account_number` - AWS Account number to associate with Aviatrix account.
- `aws_access_key` - AWS Access Key.
- `aws_role_app` - AWS App role ARN.
- `aws_role_ec2` - AWS EC2 role ARN.
- `gcloud_project_id` - GCloud Project ID.
- `gcloud_project_credentials_filepath` - GCloud Project Credentials.
- `arm_subscription_id` - Azure ARM Subscription ID.
- `arm_directory_id` - Azure ARM Directory ID.
- `arm_application_id` - Azure ARM Application ID.
- `arm_application_key` - Azure ARM Application key.
- `oci_tenancy_id` - Oracle OCI Tenancy ID.
- `oci_user_id` - Oracle OCI User ID.
- `oci_compartment_id` - Oracle OCI Compartment ID.

- `oci_api_private_key_filepath` - Oracle OCI API Private Key local file path.

# aviatrix_caller_identity

Use this data source to get the Aviatrix caller identity for use in other resources.

## Example Usage

```
# Aviatrix Caller Identity Data Source
data "aviatrix_caller_identity" "foo" {

}
```

## Argument Reference

The following arguments are supported:

- None.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `cid` - Aviatrix caller identity.

# aviatrix_firenet_vendor_integration

Use this data source to do 'save' or 'sync' for vendor integration purpose for Aviatrix FireNet.

## Example Usage

```
# Aviatrix FireNet Vendor Integration Data Source
data "aviatrix_firenet_vendor_integration" "foo" {
  vpc_id        = "vpc-abcd123"
  instance_id   = "i-09ade2592661316f8"
  vendor_type   = "Palo Alto VM Series"
  public_ip     = "10.11.12.13"
  username      = "admin"
  password      = "Avx123456#"
  firewall_name = "Avx-Firewall-Instance"
  save          = true
}
```

## Argument Reference

The following arguments are supported:

- `vpc_id` - (Required) VPC ID.
- `instance_id` - (Required) ID of Firewall instance.
- `vendor_type` - (Required) Select PAN. Valid values: "Generic", "Palo Alto VM Series", "Palo Alto VM Panorama", "Aviatrix FQDN Gateway".
- `public_ip` - (Required) The public IP address of the firewall management interface for API calls from the Aviatrix Controller.
- `username` - (Required) Firewall login name for API calls from the Controller.
- `password` - (Required) Firewall login password for API calls.
- `firewall_name` - (Optional) Name of firewall instance.
- `route_table` - (Optional) Specify the firewall virtual Router name you wish the Controller to program. If left unspecified, the Controller programs the firewall's default router.
- `save` - (Optional) Switch to save or not.
- `synchronize` - (Optional) Switch to sync or not.

# aviatrix_gateway

Use this data source to get the Aviatrix gateway for use in other resources.

## Example Usage

```
# Aviatrix Gateway Data Source
data "aviatrix_gateway" "foo" {
  account_name = "username"
  gw_name      = "gatewayname"
}
```

## Argument Reference

The following arguments are supported:

- `gw_name` - (Required) Gateway name. This can be used for getting gateway.
- `account_name` - (Optional) Account name. This can be used for logging in to CloudN console or UserConnect controller.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `account_name` - Aviatrix account name.
- `gw_name` - Aviatrix gateway name.
- `cloud_type` - Type of cloud service provider.
- `vpc_id` - VPC ID.
- `vpc_reg` - VPC Region.
- `vpc_size` - Instance type.
- `public_ip` - Public IP address of the Gateway created.

# Aviatrix R1.x Feature Changelist

## USAGE:

Tracks customer-impacting changes to Terraform environment (existing resources) throughout releases from R1.0 to present. New resources may be tracked in the Release Notes (https://www.terraform.io/docs/providers/aviatrix/guides/release-notes.html).

> **NOTE:** This changelist assumes that customers have an existing Terraform configuration and is planning to upgrade their Controller (and subsequently must also update their Aviatrix Terraform provider version to the appropriate corresponding one).

Otherwise, this list does not really apply. Please view the below list for details regarding this:

1. If a customer is transitioning to use Terraform to manage existing infrastructure, it is recommended to start from the latest Controller and Aviatrix Terraform provider version, and use the Terraform Export feature and import their infrastructure for a quick and easy migration.
    - **Please note that "Export" is still a beta feature (and only up-to-date for 4.7 at the moment)**
    - Customer can still choose to manually write their config file to their own specifications and use `terraform import` to bring their infrastructure into Terraform state management
2. If a customer is adopting Terraform for the first time, with no pre-existing infrastructure/ for managing new infrastructure, this changelist does not apply whatsoever. They simply need to follow our Terraform tutorial (https://docs.aviatrix.com/HowTos/tf_aviatrix_howto.html) to setup their Terraform environment and use the Terraform documentation featured on the left sidebar.

---

```
Last updated: R1.16 (UserConnect-4.7-patch); Terraform v0.12
```

---

## R1.0.242 (UserConnect-4.0)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | transit_vpc | enable_hybrid_connection | **No**, Terraform support added to an existing functionality on Controller. **Unless** the customer has enabled this on that transit gw through Controller, they must rectify the diff by adding that line in their config file and doing a `terraform refresh` |
| (new) | spoke_vpc | ha_gw_size | **Yes**; if customer has HA enabled for their spoke gw, they must also update their config file for ea/ spoke gw's respective sizes |
| (new) | transit_vpc | ha_gw_size | **Yes**; if customer has HA enabled for their transit gw, they must also update their config file for ea/ transit gw's respective sizes |

## R1.1.66 (UserConnect-4.1.981)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | gateway | peering_ha_eip | **No**, Terraform support added to an existing functionality on Controller. **Unless** the customer has enabled HA Peering through the Controller and manually specified the EIP of the HA GW, they must rectify the diff by adding that line in their config file and doing a `terraform refresh` |
| (deprecated) | tunnel | over_aws_peering | **Yes**, this attribute is deprecated. If users plan to continue to use AWS peering, please use the `aws_peer` resource instead |
| (new) | transit_vpc | enable_nat | **No**; Terraform support added to an existing functionality on Controller. **Unless** the customer has enabled (SNAT) on that transit gw through Controller, they must rectify the diff by adding that line in their config file and doing a `terraform refresh` |
| (new) | transit_vpc | connected_transit | **No**; Terraform support added to an existing functionality on Controller. **Unless** the customer has enabled this on that transit gw through Controller, they must rectify the diff by adding that line in their config file and doing a `terraform refresh` |
| (new) | spoke_vpc | single_az_ha | **No**; Terraform support added to an existing 3.1 feature on Controller. **Unless** the customer has enabled this on that spoke gw (instead of HA) through the Controller, they must rectify the diff by adding that line in their config file and doing a `terraform refresh` |

## R1.3.12 (UserConnect-4.1.982, 4.2.634)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | dc_extn | -- | **Yes**; this resource is deprecated. If this was being used in Terraform config files, remove it |

## R1.4.4 (UserConnect-4.2.634)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | aws_tgw | manage_vpc_attachment | **Yes**; if the `aws_tgw` resource was originally created with provider <R1.4.4 (<Controller 4.2), you must do a `terraform refresh` to update and apply the attribute's default value ("true") into the state file |

## R1.5.24 (UserConnect-4.2.764)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | gateway | dns_server | **Yes**; if customer has this in their config file, must comment it out/ remove |
| (deprecated) | transit_vpc | dns_server | **Yes**; if customer has this in their config file, must comment it out/ remove |
| (deprecated) | spoke_vpc | dns_server | **Yes**; if customer has this in their config file, must comment it out/ remove |

## R1.6.29 (UserConnect-4.2.764)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | fqdn | gw_filter_tag_list | **Yes**; new attribute implemented as a map for FQDN resource. gw_list is now implemented and nested under this attribute. From this release forward, any FQDN resource must follow the new format |
| (changed) | fqdn | gw_list | **Yes**; gateway list is now nested under `gw_filter_tag_list` |
| (new) | spoke_vpc | ha_zone | **No**; GCP support for spoke gateway added in Terraform. **Unless**, customer has GCP HA spoke gateway created through Controller that they want to import into Terraform, they must specify the `ha_zone` in that specific spoke resource in their config file, along with the originally required `ha_subnet` and `ha_gw_size` |
| (new) | spoke_vpc | vnet_and_resource_group_names | **Yes**; Azure support for spoke gateway added in Terraform. When creating an Azure spoke gateway, this attribute must be specified in place of `vpc_id`, which is used for the AWS-counterpart. **However**, note that this is deprecated in a future release (R1.10.10) and replaced with `vpc_id` |
| (new) | transit_vpc | vnet_name_resource_group | **Yes**; Azure support for transit gateway added in Terraform. When creating an Azure transit gateway, this attribute must be specified in place of `vpc_id`, which is used for the AWS-counterpart. **However**, note that this is deprecated in a future release (R1.10.10) and replaced with `vpc_id` |

## R1.8.26 (UserConnect-4.3.1275)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | gateway | ha_subnet | **Yes**; enabling traditional gateway HA has been deprecated on Controller already. However, functionality support has not yet been removed. As such, we will be removing Terraform support for enabling this feature. You will need to remove this attribute from any gateway resource that currently has gateway HA enabled through this attribute. To disable this gateway HA, customers must do so through Controller |
| (new) | gateway | peering_ha_gw_size | **Yes**; if customer has peering HA enabled for their gateway, they must also update their config file for ea/ gateway's respective sizes |
| (new) | gateway | peering_ha_zone | **No**; GCP support added for gateway. **Unless**, customer has GCP HA gateway created through Controller that they want to import into Terraform, they must specify the `peering_ha_zone` in that specific gateway resource in their config file, along with the originally required `peering_ha_gw_size`. |
| (new) | transit_vpc | insane_mode, insane_mode_az | **No**; Terraform support added to an existing 4.1 feature on Controller. **Unless**, customer has an insane mode-enabled transit gateway created through Controller that they want to import into Terraform, they must specify this attribute, along with `insane_mode_az` |
| (new) | transit_vpc | ha_insane_mode_az | **No**; see above for details. This attribute is only required if the insane mode-enabled transit gateway also has HA enabled |

## R1.9.28 (UserConnect-4.6.569)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | tunnel | cluster | **Yes**; if `cluster` was being set in this resource, remove from config file |
| (deprecated) | admin_email | -- | **Yes**; if the admin email is being set using this resource, remove from .tf file. It is no longer supported |
| (deprecated) | customer_id | -- | **Yes**; if the customer id is being set using this resource, remove from .tf file. It is no longer supported |
| (new) | site2cloud | enable_dead_peer_detection, local/remote_subnet_virtual, custom_algorithms, private_route_encryption | **No**; Terraform support added to existing S2C features on Controller. Please see site2cloud documentation for details |
| (new) | vgw_conn | enable_advertise_transit_cidr | **No**; Terraform support added to existing 4.2 feature on Controller |
| (new) | vpc | aviatrix_firenet_vpc | **No**; Terraform support added for new Firenet VPC creation. Original VPC resource added in R1.7.18 |
| (new) | transit_vpc | enable_firenet_interfaces | **No**; Terraform support added for new Firenet 4.3 feature on Controller |

## R1.10.10 (UserConnect-4.6.604)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | spoke_vpc | vnet_and_resource_group_names | **Yes**; if an Azure spoke gateway is being used, the VNet and ResourceGroup is no longer being set in this attribute. Remove and replace with `vpc_id`. This is part of the standardization in behavior across all gateway-type resources for Terraform |
| (deprecated) | transit_vpc | vnet_name_resource_group | **Yes**; if an Azure transit gateway is being used, the VNet and ResourceGroup is no longer being set in this attribute. Remove and replace with `vpc_id`. This is part of standardization in behavior across all gateway-type resources for Terraform |

## R1.11.11 (UserConnect-4.7.378)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | version | -- | **Yes**; if Controller version was being managed using this resource, remove from .tf file. This functionality has been moved to **controller_config** resource as an attribute |

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | controller_config | target_version | **No**; see above. This is an optional attribute and only needs to be set one wants/ requires Controller version management to be done through Terraform |
| (new) | vgw_conn | bgp_manual_spoke_advertise_cidrs | **No**; Terraform support added for new 4.7 feature on Controller |

## R1.12.12 (UserConnect-4.7.378)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | aws_tgw_vpn_conn | inside_ip_cidr_tun_1, pre_shared_key_tun_1/2 | **No**; Terraform support added to a new 4.6 feature on Controller. The original new **aws_tgw_vpn_conn** resource was released in R1.11.11 |

## R1.14.15 (UserConnect-4.7.474)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | gateway | max_vpn_conn | **Yes**; if any vpn-gateway was created before this release, add this attribute to .tf file(s) and specify the corresponding value(s), which may be viewed on the Controller's Gateway Page. Any new vpn-gateway created on or after this release must specify this parameter. Default is 100 |

## R1.16.20 (UserConnect-4.7.520) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (changed) | aws_tgw | security_domains, attached_vpc | **Yes**; due to Hashicorp's Terraform v0.12 release, syntactical changes were introduced. Most notably, map attributes become written as separate blocks |
| (changed) | firewall | policy | **Yes**; see above for details |
| (changed) | firewall_tag | cidr_list | **Yes**; see above for details |
| (changed) | fqdn | gw_filter_tag_list, domain_names | **Yes**; see above for details |
| (changed) | vpn_profile | policy | **Yes**; see above for details |

# Aviatrix R2.x Feature Changelist

## USAGE:

Tracks customer-impacting changes to Terraform environment (existing resources) throughout releases from R2.0 to present. New resources may be tracked in the Release Notes (https://www.terraform.io/docs/providers/aviatrix/guides/release-notes.html).

> **NOTE:** This changelist assumes that customers have an existing Terraform configuration and is planning to upgrade their Controller (and subsequently must also update their Aviatrix Terraform provider version to the appropriate corresponding one).

Otherwise, this list does not really apply. Please view the below list for details regarding this:

1. If a customer is transitioning to use Terraform to manage existing infrastructure, it is recommended to start from the latest Controller and Aviatrix Terraform provider version, and use the Terraform Export feature and import their infrastructure for a quick and easy migration.
    - **Please note that "Export" is still a beta feature (and only up-to-date for 4.7 at the moment)**
    - Customer can still choose to manually write their config file to their own specifications and use `terraform import` to bring their infrastructure into Terraform state management
2. If a customer is adopting Terraform for the first time, with no pre-existing infrastructure/ for managing new infrastructure, this changelist does not apply whatsoever. They simply need to follow our Terraform tutorial (https://docs.aviatrix.com/HowTos/tf_aviatrix_howto.html) to setup their Terraform environment and use the Terraform documentation featured on the left sidebar.

We **highly** recommend customers that are starting to adopt Terraform to manage their infrastructure, to start with our Release 2.0+ (R2.0+), which is compatible with our Controller 4.7+.

---

```
Last updated: R2.6 (UserConnect-5.1)
```

---

# R2.0 (UserConnect-4.7-patch) (Terraform v0.12)

**NOTICE:** With the Release of Aviatrix v.2, there is major restructuring of our code as well as major changes such as renaming of attributes, resources, and attribute values. All these changes are all in the name of standardization of naming conventions and resources. Although we recognize that it is a major inconvenience to customers, we believe that these changes will benefit everyone in the long-term not only for customer clarity but ease of future feature-implementations and code maintenance.

For most changes, unless stated otherwise in the tables below, after editing the respective .tf files, a simple `terraform refresh` should rectify the state of the infrastructure.

### Resource Renaming

| Diff | Resource | New Resource Name | Action Required? |
|---|---|---|---|
| (deprecated) | spoke_vpc | spoke_gateway | **Yes**; **spoke_vpc** will be deprecated and replaced with **spoke_gateway**. It will be kept for now for backward compatibility and will be removed in the future. Please use **spoke_gateway** instead<br><br>You will need to remove any pre-existing **spoke_vpc** resources from the statefile, update your .tf files to **spoke_gateway**, as well as any attributes (and corresponding values) as necessary, and `terraform import` as **aviatrix_spoke_gateway**<br><br>Attributes that are renamed include: `vpc_size`, `enable_nat`<br><br>Attributes whose accepted values are changed to booleans include: `enable_nat`, `single_az_ha` |
| (deprecated) | transit_vpc | transit_gateway | **Yes**; **transit_vpc** will be deprecated and replaced with **transit_gateway**. It will be kept for now for backward compatibility and will be removed in the future. Please use **transit_gateway** instead<br><br>You will need to remove any pre-existing **transit_vpc** resources from the statefile, update your .tf files to **transit_gateway**, as well as any attributes (and corresponding values) as necessary, and `terraform import` as **aviatrix_transit_gateway**<br><br>Attributes that are renamed include: `vpc_size`, `enable_nat`<br><br>Attributes whose accepted values are changed to booleans include: `enable_nat`, `connected_transit` |

## Attribute Renaming

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (changed) | firewall | base_allow_deny | **Yes**; renamed to `base_policy` |
| (changed) | -- | base_log_enable | **Yes**; renamed to `base_log_enabled` |
| (changed) | -- | policy: allow_deny | **Yes**; renamed to `action` |
| (changed) | -- | policy: log_enable | **Yes**; renamed to `log_enabled` |
| (changed) | fqdn | fqdn_status | **Yes**; renamed to `fqdn_enabled` |
| (changed) | gateway | vpc_size | **Yes**; change any gateways' `vpc_size` to `gw_size` |
| (changed) | gateway | vpc_net | **Yes**; change any gateways' `vpc_net` to `subnet` |
| (changed) | gateway | enable_nat | **Yes**; change any gateways' `enable_nat` to `enable_snat` |
| (changed) | spoke_vpc --> spoke_gateway | vpc_size | **Yes**, if migrating to **spoke_gateway** resource; change any **spoke_vpc** resources' `vpc_size` to `gw_size`<br><br>Please see **Resource Renaming** table for full instructions |

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (changed) | -- | enable_nat | **Yes**, if migrating to **spoke_gateway** resource; change any **spoke_vpc** resources' `enable_nat` to `enable_snat`<br><br>Please see **Resource Renaming** table for full instructions |
| (changed) | transit_vpc --> transit_gateway | vpc_size | **Yes**, if migrating to **transit_gateway** resource; change any **transit_vpc** resources' `vpc_size` to `gw_size`<br><br>Please see **Resource Renaming** table for full instructions |
| (changed) | -- | enable_nat | **Yes**, if migrating to **transit_gateway** resource; change any **transit_vpc** resources' `enable_nat` to `enable_snat`<br><br>Please see **Resource Renaming** table for full instructions |
| (changed) | tunnel | vpc_name1, vpc_name2 | **Yes**; change `vpc_name1` and `vpc_name2` to `gw_name1` and `gw_name2` respectively |

## Boolean Standardization

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (changed) | account | aws_iam | **Yes**; Accepted values are changed to **true** or **false** booleans rather than string |
| (changed) | firewall | base_log_enable | **Yes**; Accepted values are changed to **true** or **false**. Please note this attribute was also renamed. See **Attribute Renaming** table above |
| (changed) | -- | policy: log_enable | **Yes**; Accepted values are changed to **true** or **false**. Please note this attribute was also renamed. See **Attribute Renaming** table above |
| (changed) | fqdn | fqdn_status | **Yes**; Accepted values are changed to **true** or **false**. Please note this attribute was also renamed. See **Attribute Renaming** table above |
| (changed) | gateway | enable_nat | **Yes**; Accepted values are changed to **true** or **false**. Please note this attribute was also renamed. See **Attribute Renaming** table above |
| (changed) | -- | vpn_access | **Yes**; Accepted values are changed to **true** or **false** |
| (changed) | -- | enable_elb | **Yes**; see above for details |
| (changed) | -- | split_tunnel | **Yes**; see above for details |
| (changed) | -- | saml_enabled | **Yes**; see above for details |
| (changed) | -- | enable_ldap | **Yes**; see above for details |
| (changed) | -- | single_az_ha | **Yes**; see above for details |
| (changed) | -- | allocate_new_eip | **Yes**; see above for details |
| (changed) | site2cloud | ha_enabled | **Yes**; see above for details |
| (changed) | spoke_vpc --> spoke_gateway | enable_nat | **Yes**, if migrating to **spoke_gateway** resource; Accepted values are changed to **true** or **false**. Please note this attribute was also renamed. See **Attribute Renaming** table above<br><br>Please see **Resource Renaming** table for full instructions |

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (changed) | -- | single_az_ha | **Yes**, if migrating to **spoke_gateway** resource; Accepted values are changed to **true** or **false**<br><br>Please see **Resource Renaming** table for full instructions |
| (changed) | transit_vpc --> transit_gateway | enable_nat | **Yes**, if migrating to **transit_gateway** resource; Accepted values are changed to **true** or **false**. Please note this attribute was also renamed. See **Attribute Renaming** table above<br><br>Please see **Resource Renaming** table for full instructions |
| (changed) | -- | connected_transit | **Yes**, if migrating to **transit_gateway** resource; Accepted values are changed to **true** or **false**<br><br>Please see **Resource Renaming** table for full instructions |
| (changed) | tunnel | enable_ha | **Yes**; Accepted values are changed to **true** or **false** |

# R2.1 (UserConnect4.7-patch) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | transit_gateway | allocate_new_eip, eip, ha_eip | **No**; Terraform support added to feature already available for regular gateway resource. By default, `allocate_new_eip` will be set to **true**. Only works for AWS |
| (new) | spoke_gateway | allocate_new_eip, eip, ha_eip | **No**; see above for details |

# R2.3 (UserConnect5.0) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | vgw_conn | vgw_account, vgw_region | **Yes**; if customer has a **vgw_conn** resource, they must add these two attributes and their respective values to their config file and perform a `terraform refresh` |
| (new) | gateway | insane_mode, insane_mode_az, peering_ha_insane_mode_az | **No**; Terraform added support for insane mode for Azure-related resources |
| (new) | spoke_gateway, transit_gateway | insane_mode, insane_mode_az, ha_insane_mode_az | **No**; see above for details |
| (new) | aws_tgw | aviatrix_firewall, native_egress, native_firewall | **No**; Terraform support added to feature already available through Controller. These are new optional attributes for Terraform |
| (new) | spoke_gateway, transit_gateway | enable_active_mesh | **No**; Terraform support released alongside new feature available in Controller 5.0. New optional feature. Default value: **false** |

## R2.4 (UserConnect5.0) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | firewall | description | **No**; Terraform support added to feature already available for stateful firewall policies. This attribute is optional by default |
| (new) | account | oci_tenancy_id, oci_user_id, oci_compartment_id, oci_api_private_key_filepath | **No**; Terraform now supports Oracle Cloud, hence the new attributes needed to create an Oracle access account |

## R2.5 (UserConnect5.1) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (new) | gateway, spoke_gateway, transit_gateway | enable_vpc_dns_server | **No**; Terraform added support for enabling/ disabling VPC DNS server. This attribute is optional, and set 'false' by default. This feature is only available on AWS |

## R2.6 (UserConnect5.1) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (deprecated) | vgw_conn | enable_advertise_transit_cidr, bgp_manual_spoke_advertise_cidrs | **Yes**; functionality migrated over to **transit_gateway** resource. In order to maintain same functionality, customer must cut-paste the these two attributes and their respective values into the corresponding **transit_gateway** and perform a `terraform refresh` |
| (new) | transit_gateway | enable_advertise_transit_cidr, bgp_manual_spoke_advertise_cidrs | **No**; action required depends on above stated for **vgw_conn**. If customer does not have **vgw_conn** that originally advertised any sort of CIDR before this release, no action is required |
| (changed) | -- | enable_firenet_interfaces | **Yes**; if customer is using **transit_gateway** and has set `enable_firenet_interfaces` , attribute must be renamed to `enable_firenet` and a `terraform refresh` must be performed |
| (new) | -- | single_az_ha | **No**; Terraform support added for enabling/ disabling `single_az_ha` for **transit_gateway**. Unless customer has originally enabled `single_az_ha` through Controller prior to this release, then this attribute must be set to **true**, and a `terraform refresh` must be performed to rectify the state |

# Aviatrix Provider: Release Compatibility Chart

## USAGE:

Quick at-a-glance access to Aviatrix Terraform provider's release compatibility with the Controller release versions. New resources and features may be tracked in the Release Notes.

> **NOTE:** This only provides a quick glance at version compatibility between platforms. Please check the release notes for full details on new features, changes and deprecations here (https://www.terraform.io/docs/providers/aviatrix/guides/release-notes.html).

`Last updated: R2.6 (UserConnect-5.1)`

| Terraform Version (v) | Aviatrix Provider Version (R) | Supported Controller Version |
|---|---|---|
| 0.11 | 1.0 | UserConnect-4.0 |
| 0.11 | 1.1 | UserConnect-4.1 |
| 0.11 | 1.2 | UserConnect-4.1 |
| 0.11 | 1.3 | UserConnect-4.2 |
| 0.11 | 1.4 | UserConnect-4.2 |
| 0.11 | 1.5 | UserConnect-4.2 |
| 0.11 | 1.6 | UserConnect-4.2 |
| 0.11 | 1.7 | UserConnect-4.3 |
| 0.11 | 1.8 | UserConnect-4.3 |
| 0.11 | 1.9 | UserConnect-4.6 |
| 0.11 | 1.10 | UserConnect-4.6 |
| 0.11 | 1.11 | UserConnect-4.7 |
| 0.11 | 1.12 | UserConnect-4.7 |
| 0.11 | 1.13 | UserConnect-4.7 |
| 0.11 | 1.14 | UserConnect-4.7 |
| 0.11 | 1.15 | UserConnect-4.7 |
| **0.12** [1] | 1.16 | UserConnect-4.7 |
| 0.12 | **2.0** [2] | UserConnect-4.7 |
| 0.12 | 2.1 | UserConnect-4.7 |
| 0.12 | 2.2 | UserConnect-4.7 |
| 0.12 | 2.3 | UserConnect-5.0 |

| Terraform Version (v) | Aviatrix Provider Version (R) | Supported Controller Version |
|---|---|---|
| 0.12 | 2.4 | UserConnect-5.0 |
| 0.12 | 2.5 | UserConnect-5.1 |
| 0.12 | 2.6 | UserConnect-5.1 |

[1] : Note that Terraform v0.12 is not backwards-compatible with previous Terraform versions. For R1.16, there will be a need to change some syntax in the Terraform configuration files. Please see Hashicorp's announcement (https://www.hashicorp.com/blog/announcing-terraform-0-12) for more information

[2] : With R2.0, there is major code restructuring that includes attribute/resource renaming and changes to attribute values. We *highly* recommend customers reference the R2.0 upgrade guide (https://www.terraform.io/docs/providers/aviatrix/guides/v2-upgrade-guide.html) for detailed instructions before upgrading to R2.0

## Example:

If your Aviatrix Controller is on UserConnect-5.0.x, you should be using Hashicorp Terraform v0.12.x and our Aviatrix provider R2.4.

Although R2.3 is also compatible with UserConnect-5.0, we recommend using the latest compatible provider version corresponding to the Controller.

```
provider "aviatrix" {
  controller_ip           = "1.2.3.4"
  username                = "admin"
  password                = "password"
  skip_version_validation = false
  version                 = "2.4.0"
}
```

# Aviatrix Provider: Release Notes

## USAGE:

Track all Aviatrix Terraform provider's releases. New resources, features, and bug fixes will be tracked here.

```
Last updated: R2.6 (UserConnect-5.1)
```

## 2.6.0

### Notes:

- Supported Controller version: **UserConnect-5.1.935**
- Supported Terraform version: **v0.12.x**

### Features:

1. New resources:
    - **aviatrix_firewall_instance**
    - **aviatrix_firenet**
2. New data source:
    - **aviatrix_firenet_vendor_integration**
3. Added support to create security domain of `aviatrix_firewall`, `native_egress` or `native_firewall` in **aviatrix_aws_tgw**
4. Added support to attach/detach FireNet VPC to/from TGW in **aviatrix_aws_tgw**

### Enhancements:

1. Separated `subnets` to `public_subnets` and `private_subnets` in **aviatrix_vpc**
2. Moved `enable_advertise_transit_cidr` and `bgp_manual_spoke_advertise_cidrs` from **aviatrix_vgw_conn** to **aviatrix_transit_gateway**, and made **aviatrix_vgw_conn** non-updatable
3. Added option to use `byol` for test-infra, and updated test-infra to support acceptance test for new resources and data sources
4. Added err body printing for the err that can not decode output of rest api
5. Renamed `enable_firenet_interfaces` to `enable_firenet` in **aviatrix_transit_gateway**

6. Added option to enable/disable `single_az_ha` in **aviatrix_transit_gateway**

### Bug Fixes:

1. Fixed issue where updating **aviatrix_account**'s `aws_account_number` causes crash

## 2.5.0

### Notes:

- Release date: **(2 Oct 2019)**
- Supported Controller version: **UserConnect-5.1.738**
- Supported Terraform version: **v0.12.x**

### Features:

1. Added support for enabling/ disabling vpc_dns_server ( `enable_vpc_dns_server` ) under the AWS (Amazon Web Services) cloud provider for the following resources:
    - **aviatrix_gateway**
    - **aviatrix_spoke_gateway**
    - **aviatrix_transit_gateway**

### Enhancements:

1. Implemented a shell script tool to export test-infra output for acceptance test

## 2.4.0

### Notes:

- Release date: **(27 Sep 2019)**
- Supported Controller version: **UserConnect-5.0.2761**
- Supported Terraform version: **v0.12.x**

### Features:

1. Added support for OCI (Oracle Cloud Infrastructure) in the following resources:

    - **aviatrix_account**
    - **aviatrix_gateway**
    - **aviatrix_spoke_gateway**
    - **aviatrix_transit_gateway**
2. Added support for GCP (Google Cloud Platform) in **aviatrix_transit_gateway**
3. Updated test-infra to support acceptance test for OCI

### Enhancements:

1. Added `description` as an attribute under policy in **aviatrix_firewall**

### Bug Fixes:

1. Fixed issue where HA gateway could not be deleted before the primary gateway for GCP transit gateway

## 2.3.36

### Notes:

- Release date: **(16 Sep 2019)**
- Supported Controller version: **UserConnect-5.0.2675**
- Supported Controller version: **v0.12.x**

### Bug Fixes:

1. Fixed acceptance test cases

## 2.3.35

### Notes:

- Release date: **(10 Sep 2019)**
- Supported Controller version: **UserConnect-5.0.2632**
- Supported Terraform version: **v0.12.x**

### Features

1. Added support for Insane Mode for ARM (Azure Resource Manager) in the following resources:
    - **aviatrix_gateway**
    - **aviatrix_spoke_gateway**
    - **aviatrix_transit_gateway**
2. Added support for `vgw_account` and `vgw_region` in **aviatrix_vgw_conn**
3. Added support for creating `aviatrix_firewall`, `native_egress`, and `native_aviatrix` domain in **aviatrix_aws_tgw**
4. Added support for ActiveMesh mode for the following resources:
    - **aviatrix_gateway**
    - **aviatrix_spoke_gateway**
    - **aviatrix_transit_gateway**

### Enhancements

1. Added `subnet_id` as an output attribute for **aviatrix_vpc**
2. Added support to edit `vpn_cidr` by gateway instead of just load balancer

### Bug Fixes:

1. Fixed enabling/ disabling advertising CIDRs issue in **aviatrix_vgw_conn**

## 2.2.0

### Notes:

- Release date: **(30 Aug 2019)**
- Supported Controller version: **UserConnect-4.7.591**
- Supported Terraform version: **v0.12.x**
- Initial Release for Official provider to allow: `terraform init` setup

## 2.1.29

### Notes:

- Release date: **(19 Aug 2019)**
- Supported Controller version: **UserConnect-4.7.585**
- Supported Terraform version: **v0.12.x**

### Features:

1. Added support for specifying EIP ( `allocate_new_eip` , `eip` , `ha_eip` ) of the primary and HA gateway under the AWS (Amazon Web Services) cloud provider for the following resources:
    - **spoke_gateway**
    - **transit_gateway**
2. Added new resource: **aviatrix_saml_endpoint**. Currently only supports text IDP metadata type

## 2.0.36

### Notes:

- Release date: **(25 Jul 2019)**
- Supported Controller version: **UserConnect-4.7.520**
- Supported Terraform version: **v0.12.x**

Major code-base restructuring, featuring renaming of attributes, resources, and attribute values. All these changes are all in the name of standardization of naming conventions and resources

### Changes:

Please see the R2.0 feature changelist table (https://www.terraform.io/docs/providers/aviatrix/guides/feature-changelist-v2.html#r2-0-userconnect-4-7-patch-terraform-v0-12-) for full details on the changes

## 1.16.20

### Notes:

- Release date: **(25 Jul 2019)**
- Supported Controller version: **UserConnect-4.7.520**
- Supported Terraform version: **v0.12.x**
- Updated R1.x Feature Changelist

### Enhancements:

1. Now supports Terraform v0.12.x
2. Now uses Go Mod

## 1.15.05

### Notes:

- Release date: **(15 Jul 2019)**
- Supported Controller version: **UserConnect-4.7.474**
- Supported Terraform version: **v0.11.x**
- Updated R1.x Feature Changelist

### Enhancements:

1. Added 10s sleep time before updating `split_tunnel` for VPN gateway creation
2. Updated test-infra

## 1.14.15

### Notes:

- Release date: **(11 Jul 2019)**
- Supported Controller version: **UserConnect-4.7.474**
- Supported Terraform version: **v0.11.x**
- Updated R1.x Feature Changelist

### Features:

1. Added support for `max_vpn_conn` in **aviatrix_gateway** resource

## 1.13.14

### Notes:

- Release date: **(28 Jun 2019)**
- Supported Controller version: **UserConnect-4.7.419**
- Supported Terraform version: **v0.11.x**
- Added R1.x Feature Changelist

### Enhancements:

1. Added defer function for the following resources:
    - **aviatrix_aws_tgw**
    - **aviatrix_fqdn**
    - **aviatrix_spoke_vpc**
    - **aviatrix_transit_vpc**
    - **aviatrix_site2cloud**
    - **aviatrix_vgw_conn**
2. Added test-infra for Hashicorp acceptance

## 1.12.12

### Notes:

- Release date: **(20 Jun 2019)**
- Supported Controller version: **UserConnect-4.7.378**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added support for inside IP CIDR and pre-shared key for tunnel1 and tunnel2 of **aviatrix_aws_tgw_vpn_conn**
    - `inside_ip_cidr_tun_1`
    - `inside_ip_cidr_tun_2`
    - `pre_shared_key_tun_1`
    - `pre_shared_key_tun_2`

### Enhancements:

1. Added defer function for **aviatrix_gateway**

# 1.11.16

## Notes:

- Release date: **(18 Jun 2019)**
- Supported Controller version: **UserConnect-4.7.378**
- Supported Terraform version: **v0.11.x**

## Features:

1. Added support for `bgp_manual_spoke_advertise_cidrs` for **aviatrix_vgw_conn** resource
2. Added new resource **aviatrix_vpn_user_accelerator** to support VPN user acceleration through Terraform
3. Added new resource **aviatrix_aws_tgw_vpn_conn** to support attaching/ detaching VPN to TGWs

## Enhancements:

1. Deprecated **version** resource, and changed to an attribute `target_version` under **aviatrix_controller_config** to consolidate controller configuration behaviors under one resource

# 1.10.10

## Notes:

- Release date: **(7 Jun 2019)**
- Supported Controller version: **UserConnect-4.6.604**
- Supported Terraform version: **v0.11.x**

## Enhancements:

1. Deprecated `vnet_and_resource_group_names` and `vnet_name_resource_group` in **aviatrix_spoke_vpc** and **aviatrix_transit_vpc**, respectively and replaced with `vpc_id` in order to standardize attributes across various cloud providers

# 1.9.28

## Notes:

- Release date: **(3 Jun 2019)**
- Supported Controller version: **UserConnect-4.6.569**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added the following support for **aviatrix_site2cloud**:
    - private route encryption ( `private_route_encryption` )
    - custom algorithm ( `custom_algorithms` )
    - SSL server pool for TCP tunnel types ( `ssl_server_pool` )
    - dead peer detection ( `enable_dead_peer_detection` )
2. Added support for advertising transit CIDRs ( `enable_advertise_transit_cidr` ) for **aviatrix_vgw_conn**
3. Added support creating an Aviatrix FireNet VPC ( `aviatrix_firenet_vpc` ) for **aviatrix_vpc**
4. Added support for enabling a transit gateway for Aviatrix FireNet; ( `enable_firenet_interfaces` ) in **aviatrix_transit_vpc**

### Enhancements:

1. Deprecated the following resources to consolidate workflow:
    - **aviatrix_admin_email**
    - **aviatrix_customer_id**
2. Deprecated `cluster` from **aviatrix_tunnel** resource due to being a deprecated feature in the Controller

## 1.8.26

### Notes:

- Release date: **(30 May 2019)**
- Supported Controller version: **UserConnect-4.3.1275**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added support for configuring gateway size for peering HA gateway ( `peering_ha_gw_size` ) for **aviatrix_gateway**
2. Added Insane Mode support ( `insane_mode` , `insane_mode_az` ) for **aviatrix_transit_vpc**
3. Added support for GCP (Google Cloud Platform) in **aviatrix_gateway**

4. Added new resource **aviatrix_arm_peer** to support ARM (Azure Resource Manager) VNet peering
5. Added acceptance test support for import feature for all resources

### Enhancements:

1. Deprecated `ha_subnet` from **aviatrix_gateway**

## 1.7.18

### Notes:

- Release date: **(9 May 2019)**
- Supported Controller version: **UserConnect-4.3.1253**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added new resource **aviatrix_vpc** to support Controller's Create VPC Tool to create easily create VPCs, subnets
2. Added support for "mapped" connection types ( `connection_type` ) in **aviatrix_site2cloud**

### Enhancements:

1. Set supportedVersion as a global variable
2. Updated GetVPNUser to call get_vpn_user_by_name instead of list_vpn_user

### Bug Fixes:

1. Fixed **aviatrix_site2cloud**'s `connection_type` read/ refresh issue
2. Fixed **aviatrix_vgw_conn** read/ refresh/ import issue

## 1.6.29

### Notes:

- Release date: **(3 May 2019)**
- Supported Controller version: **UserConnect-4.2.764**

- Supported Terraform version: **v0.11.x**

### Features:

1. Added ARM (Azure Resource Manager) and GCP (Google Cloud Platform) for **aviatrix_spoke_vpc**
2. Added ARM support for **aviatrix_transit_vpc**
3. Added support for FQDN source IP filtering `source_ip_list` in **aviatrix_fqdn** resource
4. Added migration support for **aviatrix_aws_tgw** resource
5. Added **aviatrix_controller_config** resource that supports the following features:
    - system-wide FQDN exception rule ( `fqdn_exception_rule` )
    - security group management ( `security_group_management` )
    - http access ( `http_access` )

### Enhancements:

1. Added controller version checking in the provider to ensure compatibility between Aviatrix Terraform provider and Controller

## 1.5.24

### Notes:

- Release date: **(15 Apr 2019)**
- Supported Controller version: **UserConnect-4.2.764**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added ARM (Azure Resource Manager) and GCP (Google Cloud Platform) support for **aviatrix_account**

### Enhancements:

1. Moved goaviatrix library from vendor to root folder
2. Deprecated `dns_server` for the following resources:
    - **aviatrix_gateway**
    - **aviatrix_spoke_vpc**
    - **aviatrix_transit_vpc**

3. Added description for all attributes
4. Added import support for **aviatrix_gateway**'s `split_tunnel`

### Bug Fixes:

1. Fixed migration/ update issue for `manage_vpc_attachment` in **aviatrix_aws_tgw** resource
2. Fixed failing to destroy **aviatrix_vgw_conn** despite being destroyed in Controller UI
3. Fixed refresh issue for deleted **aviatrix_fqdn** through Controller UI
4. Fixed read/ refresh issue for **aviatrix_site2cloud** where resource count exceeds 3

## 1.4.4

### Notes:

- Release date: **(28 Mar 2019)**
- Supported Controller version: **UserConnect-4.2.634**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added new resource **aviatrix_aws_tgw_vpc_attachment** to simplify/ add an option on how users can choose to manage attaching/ detaching VPCs to and from their **aviatrix_aws_tgw**

### Enhancements:

1. Updated **aviatrix_aws_tgw** to allow creation of only the TGW, as well as allowing management of VPC attachments to be done either within the resource, or though **aviatrix_aws_tgw_vpc_attachment**
2. updated documentation for **aviatrix_aws_peer** resource
3. updated **aviatrix_fqdn** to block updating `fqdn_tag`

## 1.3.12

### Notes:

- Release date: **(21 Mar 2019)**
- Supported Controller version: **UserConnect-4.1.982** and **4.2.634**

- Supported Terraform version: **v0.11.x**

### Enhancements:

1. Deprecated **aviatrix_dc_extn** resource due to removed support from Controller
2. Added version information

### Bug Fixes:

1. Fixed **aviatrix_firewall**'s `base_allow_deny` on refresh
2. Fixed **aviatrix_site2cloud**'s refresh, update and import issues
3. Fixed **aviatrix_aws_peer**'s refresh, update and import issues

## 1.2.12

### Notes:

- Release date: **(15 Mar 2019)**
- Supported Controller version: **UserConnect-4.1.981**
- Supported Terraform version: **v0.11.x**

### Changes:

1. Temporarily reverted refresh changes for the following resources:
    - **aviatrix_aws_peer**
    - **aviatrix_site2cloud**

### Bug Fixes:

1. Fixed **aviatrix_site2cloud** to ignore `local_subnet_cidr` changes

## 1.2.10

> **NOTE:** This release is unsupported and deprecated

### Notes:

- Release date: **(14 Mar 2019)**
- Supported Controller version: **UserConnect-4.1.981**
- Supported Terraform version: **v0.11.x**

### Bug Fixes:

1. Fixed `tag_list` reordering issue on **aviatrix_gateway**
2. Fixed refresh issues for the following resources:
    - **aviatrix_aws_peer**
    - **aviatrix_site2cloud**
    - **aviatrix_vgw_conn**

## 1.1.66

### Notes:

- Release date: **(6 Mar 2019)**
- Supported Controller version: **UserConnect-4.1.981**
- Supported Terraform version: **v0.11.x**

### Features:

1. Added support for specifying EIP ( `peering_ha_eip` ) for the HA gateway in **aviatrix_gateway** resource
2. All resources now support `terraform import`

### Enhancements:

1. Enhanced returned error messages to show REST API names
2. Deprecated `over_aws_peering` in **aviatrix_tunnel** resource
3. Enhanced refresh functionality for the following resources:
    - **aviatrix_aws_tgw**
    - **aviatrix_admin_email**
4. **aviatrix_firewall** resource enhanced to have policy validation

### Bug Fixes:

1. Fixed URL encode error for all resources
2. Fixed port requirement for ICMP protocol in **aviatrix_fqdn**
3. Fixed **aviatrix_transit_vpc** resource to support empty `tag_list`
4. Fixed **aviatrix_vpn_user** re-ordering issue on refresh

# 1.0.242

## Notes:

- Release date: **(26 Feb 2019)**
- Supported Controller version: **UserConnect-4.1.981**
- Supported Terraform version: **v0.11.x**

## Features:

1. Support for Terraform's `create`, `destroy`, `refresh`, `update`, and acceptance tests for most of the following resources:
    - **data_source_aviatrix_account**
    - **data_source_aviatrix_caller_identity**
    - **data_source_aviatrix_gateway**
    - **resource_aviatrix_account**
    - **resource_aviatrix_account_user**
    - **resource_aviatrix_admin_email**
    - **resource_aviatrix_aws_peer**
    - **resource_aviatrix_aws_tgw**
    - **resource_aviatrix_customer_id**
    - **resource_aviatrix_dc_extn**
    - **resource_aviatrix_firewall**
    - **resource_aviatrix_firewall_tag**
    - **resource_aviatrix_fqdn**
    - **resource_aviatrix_gateway**
    - **resource_aviatrix_site2cloud**
    - **resource_aviatrix_spoke_vpc**

- resource_aviatrix_transit_gateway_peering
- resource_aviatrix_transit_vpc
- resource_aviatrix_transitive_peering
- resource_aviatrix_tunnel
- resource_aviatrix_version
- resource_aviatrix_vgw_conn
- resource_aviatrix_vpn_profile
- resource_aviatrix_vpn_user

# Aviatrix Terraform Provider Upgrade Guide (R1.x to R2.x)

## USAGE:

For customers who are currently already using Terraform to manage their infrastructure prior to Controller 4.7 and are looking to upgrade their Controller to 4.7.501+ due to Field Notice 005 or other reasons, please follow along for guidance on the upgrade process to ensure a smooth transition.

---

In summary:

**Current Setup/ Configuration:**

- Controller <4.7
- Terraform v0.11.x
- Aviatrix Terraform provider R1.xx
- Looking to upgrade Controller to 4.7+

---

### !! Important !!

As Aviatrix is now an official Terraform provider, the process to setup and upgrade between versions have been significantly simplified. Upgrading to newer releases is as simple as specifying the "version" in the Aviatrix "provider" block, and running a `terraform init`. Customers no longer need to source from Github and build our provider locally.

Customers who previously have used Aviatrix as a provider prior to official acceptance may transition to the official provider by following Step 5.2 in our setup tutorial (https://docs.aviatrix.com/HowTos/tf_aviatrix_howto.html#troubleshooting).

For any further questions, please check out our Support Center (https://docs.aviatrix.com/Support/support_center_terraform.html).

---

## Context

With the Aviatrix Terraform provider R2.0, there is major restructuring of our code as well as major changes such as renaming of attributes, resources, and attribute values. All these changes are all in the name of standardization of naming conventions and resources. Although we recognize that it is a major inconvenience to customers, we believe that these changes will benefit everyone in the long-term not only for customer clarity but ease of future feature-implementations and code maintenance.

For most changes, unless stated otherwise, after editing the respective (.tf) files, a simple `terraform refresh` should rectify the state of the infrastructure.

> **NOTE:** For future releases, Aviatrix Terraform provider R2.3+ is only compatible with Terraform v0.12 and Aviatrix Controller 5.0+.

In this document, we broke down the upgrade process in multiple phases to make it easier to follow along.

In summary:

| Phase | Notes | From (Controller / Provider / Terraform) | To (Controller / Provider / Terraform) | Effort |
|---|---|---|---|---|
| 1 | Upgrade Controller | <4.7 / R1.xx / v0.11.x | **4.7** / R1.xx / v0.11.x | low |
| 2 | Upgrade Terraform | 4.7 / R1.xx / v0.11.x | 4.7 / R1.xx / **v0.12.x** | low |
| 3 | Upgrade Provider | 4.7 / R1.xx / v0.12.x | 4.7 / **R2.0** / v0.12.x | non-trivial |
| FUTURE | Upgrade both | 4.7 / R2.0 / v0.12.x | **5.0+** / **R2.0+** / v0.12.x | low |

# Phase 1: Upgrading Aviatrix Controller to 4.7

**Summary:** This phase involves first upgrading the Controller to 4.7. Afterwards, customers must update their (.tf) files as necessary, then upgrade their Aviatrix Terraform provider version, and then perform `terraform refresh` to rectify their state.

1. Upgrade Controller through Aviatrix Console Web GUI. Please see below links for references:
    - Controller Upgrade documentation (https://docs.aviatrix.com/HowTos/inline_upgrade.html)
    - Controller Release Notes (https://docs.aviatrix.com/HowTos/UCC_Release_Notes.html)
2. Update Terraform files (.tf) as necessary. Please reference documentation linked below to note specific changes to any resource attributes that you may be using in your configuration:
    - Please refer **up to** R1.14's table (https://www.terraform.io/docs/providers/aviatrix/guides/feature-changelist.html#r1-14-15-userconnect-4-7-474-) (for Terraform 0.11) (compatible with Controller 4.7):
    - Aviatrix Terraform Provider Release Notes (https://www.terraform.io/docs/providers/aviatrix/guides/release-notes.html)
3. Update Aviatrix Terraform provider:
    - Navigate to Aviatrix Terraform provider directory:
    - Mac:
        - `cd $GOPATH/src/github.com/terraform-providers`
    - Windows:
        - `cd %GOPATH%\src\github.com\terraform-providers\terraform-provider-aviatrix`
    - Update repository by pulling changes:
    - `git pull`
    - Change branch to **UserConnect-4.7-TF.11**:
    - `git checkout UserConnect-4.7-TF.11`
    - Build the provider:

- Mac:
  - `make fmt`
  - `make build`
- Windows:
  - `go fmt`
  - `go install`

4. Navigate to your Terraform directory/ directories and refresh Terraform state to update:
   - `terraform init` - to reinitialise a working Terraform environment for the current directory
   - `terraform refresh`
   - Please perform a plan command to note any deltas:
   - `terraform plan`
   - If there are any deltas still, you may fix and run refresh/ plan again accordingly

## Example Walk-through

Say you are upgrading from Controller 4.1, Terraform v0.11, Aviatrix Terraform provider R1.1. And for example, you are using Terraform to manage your topology consisting of single AWS VPN gateway that you use to connect VPN users. Your Terraform file(s) might look something like this:

```
# VPN_setup.tf

# ...

resource "aviatrix_gateway" "aws_vpn_gw" {
  cloud_type    = 1
  account_name  = "devops"
  gw_name       = "aws_vpn_gw"
  vpc_id        = "vpc-abc123"
  vpc_reg       = "us-east-1"
  vpc_size      = "t2.micro"
  vpc_net       = "10.0.0.0/24"

  dns_server    = "8.8.8.8"
  vpn_access    = "yes"
  vpn_cidr      = "192.168.43.0/24"
  enable_elb    = "yes"
  elb_name      = "example-elb-name"

  # ...

}

resource "aviatrix_vpn_user" "vpn_user_1" {
  # ...
}

# ...
```

According to the table **R1.5** for UserConnect-4.2, the attribute `dns_server` for gateway resource is deprecated and to remove it from your file(s) if you are using it. In addition, according to the table **R1.14** for UserConnect-4.7, the attribute `max_vpn_conn` is a new and required for any vpn-gateway from this release moving forward. So in summary, in this example, you will have to remove `dns_server` and add `max_vpn_conn` into your gateway resource. Your file(s) should now look something like this (only relevant section shown):

```
# VPN_setup.tf

# ...

resource "aviatrix_gateway" "aws_vpn_gw" {
  cloud_type    = 1
  account_name  = "devops"
  # ...
  vpc_net       = "10.0.0.0/24"

  # note that dns_server is removed
  vpn_access    = "yes"
  vpn_cidr      = "192.168.43.0/24"
  max_vpn_conn  = 100 # newly added
  enable_elb    = "yes"
  elb_name      = "example-elb-name"

  # ...

}

# ...
```

After completion of editing your Terraform files, the rest of this phase is as simple as updating your Aviatrix Terraform provider repository and using the correct corresponding release.

Navigate to your local Aviatrix Terraform provider repository, which by default, if setup according to our initial setup doc here (https://github.com/terraform-providers/terraform-provider-aviatrix/blob/master/README.md), is `$GOPATH/src/github.com/terraform-providers` . We will then use Git to pull latest changes from our remote repository ( `git pull` ), and switch to the branch that corresponds with our Controller version ( `git checkout UserConnect-4.7-TF.11` ). Build the provider as according to Step 3 depending on your OS. Finally, navigate back to where your Terraform files reside, and perform a `terraform init` to reinitialise the Terraform environment for the current directory based on the new provider, and run a `terraform refresh` . A `terraform plan` may be performed to catch any deltas. If there are still any deltas, you may fix and repeat the refresh/ plan steps again.

---

# Phase 2: Upgrading Hashicorp's Terraform to v0.12

**Summary:** This phase involves upgrading Hashicorp's Terraform from v0.11 to v0.12. As far as the Aviatrix Terraform provider is concerned, Hashicorp's Terraform's v0.12 only involves syntactical changes. In addition, customers must also update their (.tf) files as necessary, then upgrade their Aviatrix Terraform provider version, and then perform `terraform refresh` to rectify their state.

1. Upgrade Terraform version to v0.12:
    - Hashicorp Terraform Upgrade documentation (https://www.terraform.io/upgrade-guides/0-12.html)

- Hashicorp Terraform 0.12 Release Notes (https://www.hashicorp.com/blog/announcing-terraform-0-12)

2. Update Terraform files (.tf) as necessary. Please reference documentation linked below to note specific changes to any resource attributes that you may be using in your configuration:
   - Please follow the **R1.16's table** (https://www.terraform.io/docs/providers/aviatrix/guides/feature-changelist.html#r1-16-20-userconnect-4-7-520-terraform-v0-12-) (for Terraform 0.12) (compatible with Controller 4.7):
   - The same table is shown below for your convenience:

## R1.16.20 (UserConnect-4.7.520) (Terraform v0.12)

| Diff | Resource | Attribute | Action Required? |
|---|---|---|---|
| (changed) | aws_tgw | security_domains, attached_vpc | **Yes**; due to Hashicorp's Terraform v0.12 release, syntactical changes were introduced. Most notably, map attributes become written as separate blocks |
| (changed) | firewall | policy | **Yes**; see above for details |
| (changed) | firewall_tag | cidr_list | **Yes**; see above for details |
| (changed) | fqdn | gw_filter_tag_list, domain_names | **Yes**; see above for details |
| (changed) | vpn_profile | policy | **Yes**; see above for details |

1. Update Aviatrix Terraform provider:
   - Navigate to Aviatrix Terraform provider directory:
   - Mac:
     - `cd $GOPATH/src/github.com/terraform-providers`
   - Windows:
     - `cd %GOPATH%\src\github.com\terraform-providers\terraform-provider-aviatrix`
   - Update repository by pulling changes:
   - `git pull`
   - Change branch to **UserConnect-4.7-TF.12-v1**:
   - `git checkout UserConnect-4.7-TF.12-v1`
   - Build the provider:
   - Mac:
     - `make fmt`
     - `make build`
   - Windows:
     - `go fmt`
     - `go install`

2. Navigate to your Terraform directory/ directories and refresh Terraform state to update:

- `terraform init` - to reinitialise a working Terraform environment for the current directory
- `terraform refresh`
- Please perform a plan command to note any deltas:
- `terraform plan`
- If there are any deltas still, you may fix and run refresh/ plan again accordingly

## Example Walk-through

As stated previously, as far as the Aviatrix Terraform provider is concerned, Hashicorp's Terraform's v0.12 only involves syntactical changes in regards to certain resources. You will need to first follow Hashicorp's upgrade instructions (https://www.terraform.io/upgrade-guides/0-12.html) as seen in Step 1 of this phase; please see the Hashicorp Terraform v0.12 release notes (https://www.hashicorp.com/blog/announcing-terraform-0-12) as well. Upon doing so, you will need to update your Terraform file(s) to accommodate for the new syntactical changes.

For example, if your Terraform manages a VPN-configuration that includes a vpn-gateway, vpn-users and some profiles, it might currently look something like this in v0.11:

```
# v0.11 Example .tf file
# ...

# VPN-related resources
resource "aviatrix_gateway" "vpn-gw" {
  # ...
}

resource "aviatrix_vpn_user" "vpn-user-1" {
  # ...
}

resource "aviatrix_vpn_profile" "vpn-profile-1" {
  name      = "vpn_profile_1"
  base_rule = "allow_all"
  users     = ["vpn-user-1"]
  policy = [
    {
      action  = "deny"
      proto   = "tcp"
      port    = "443"
      target  = "10.0.0.0/32"
    },
    {
      action  = "deny"
      proto   = "tcp"
      port    = "443"
      target  = "10.0.0.1/32"
    }
  ]
  depends_on = ["aviatrix_vpn_user.vpn-user-1"]
}
```

With the new Terraform v0.12, the vpn_profile resource, along with others, as documented in the **R1.16** table above, the map attributes are now written as separate blocks, as seen below (only relevant section shown):

```
# v0.12 Example .tf file
# ...

# VPN-related resources
# ...

resource "aviatrix_vpn_profile" "vpn-profile-1" {
  name      = "vpn_profile_1"
  base_rule = "allow_all"
  users     = ["vpn-user-1"]

  # Note the change: each policy is written as a separate block now
  policy {
    action  = "deny"
    proto   = "tcp"
    port    = "443"
    target  = "10.0.0.0/32"
  }
  policy {
    action  = "deny"
    proto   = "tcp"
    port    = "443"
    target  = "10.0.0.1/32"
  }
  depends_on = ["aviatrix_vpn_user.vpn-user-1"]
}
```

Once again, as in Phase 1, you will have to update your Aviatrix Terraform provider repository and use the correct corresponding release.

Navigate to your local Aviatrix Terraform provider repository, which by default, if setup according to our initial setup doc here (https://github.com/terraform-providers/terraform-provider-aviatrix/blob/master/README.md), is `$GOPATH/src/github.com/terraform-providers` . We will then use Git to pull latest changes from our remote repository ( `git pull` ), and switch to the branch that corresponds with our Controller version ( `git checkout UserConnect-4.7-TF.12-v1` ). Build the provider as according to Step 3 depending on your OS. Finally, navigate back to where your Terraform files reside, and perform a `terraform init` to reinitialise the Terraform environment for the current directory based on the new provider, and run a `terraform refresh` . A `terraform plan` may be performed to catch any deltas. If there are still any deltas, you may fix and repeat the refresh/ plan steps again.

---

## Phase 3: Upgrading Aviatrix Terraform Provider to R2.0

**Summary:** This will be the largest phase in terms of the upgrade process. While this phase only involves upgrading the customer's (.tf) files as necessary, the amount of changes from R1.xx to R2.0+ are not small. Afterwards, customers must upgrade their Aviatrix Terraform provider version, and then perform `terraform refresh` or `terraform import` as necessary.

1. Update Terraform files (.tf) as necessary. Please reference documentation linked below to note specific changes to any resource attributes that you may be using in your configuration:

- Aviatrix Provider R2.x Feature Changelist (https://www.terraform.io/docs/providers/aviatrix/guides/feature-changelist-v2.html)

2. Update Aviatrix Terraform provider:
   - Navigate to Aviatrix Terraform provider directory:
   - Mac:
     - `cd $GOPATH/src/github.com/terraform-providers`
   - Windows:
     - `cd %GOPATH%\src\github.com\terraform-providers\terraform-provider-aviatrix`
   - Update repository by pulling changes:
   - `git pull`
   - Change branch to **UserConnect-4.7-TF.12-v2**:
   - `git checkout UserConnect-4.7-TF.12-v2`
   - Build the provider:
   - Mac:
     - `make fmt`
     - `make build`
   - Windows:
     - `go fmt`
     - `go install`
3. Navigate to your Terraform directory/ directories and refresh/ import to update:
   - `terraform init` - to reinitialise a working Terraform environment for the current directory
   - **Rules to determine whether to refresh or import:**
   - **NOTE:** Rules are listed in terms of priority. Import takes precedence over refresh; meaning if your resource has one of the below import rules apply, that resource will require an import no matter how many of the refresh rules apply to it
   - If you are using **transit_vpc** or **spoke_vpc** resources, note they are deprecated and support will eventually be removed. **transit_gateway** and **spoke_gateway** will replace them respectively. `terraform import` for these resources must be done to rectify the state
     - **Please refer to the documentation in step 1 for more detailed instructions**
   - If your resource uses an attribute whose accepted values changed, `terraform import` must be done to rectify the state
   - If your resource uses an attribute that has been renamed, `terraform refresh` is sufficient to rectify the state
   - If your resource uses an attribute that has been renamed **AND** had its accepted values changed, `terraform refresh` is sufficient to rectify the state

## Example Walk-through

Phase 3 will definitely be a larger task, but is still nothing too different from what has been done in the previous 2 phases. Here we will use an example to demonstrate the refresh/ import rules more clearly.

Let's say you have finished Phase 2, and are currently on Controller 4.7, Terraform v0.12 and now you need to upgrade your Aviatrix Terraform provider to R2.0. Let's take the completed vpn-configuration example from Phase 1:

```
# VPN_setup.tf

# ...

resource "aviatrix_gateway" "aws_vpn_gw" {
  cloud_type    = 1
  account_name  = "devops"
  gw_name       = "aws_vpn_gw"
  vpc_id        = "vpc-abc123"
  vpc_reg       = "us-east-1"
  vpc_size      = "t2.micro"
  vpc_net       = "10.0.0.0/24"

  # note that dns_server is removed
  vpn_access    = "yes"
  vpn_cidr      = "192.168.43.0/24"
  max_vpn_conn  = 100 # newly added
  enable_elb    = "yes"
  elb_name      = "example-elb-name"

  enable_nat    = "yes"
  # ...

}
```

We will now need to update our Terraform file(s). In this case, note that due to the attribute re-naming, we will have to rename attributes such as `vpc_size` and `vpc_net` to `gw_size` and `subnet`, respectively, as it is much more clear as to what these attributes refer to. `enable_nat` must also be changed to `enable_snat` here. In addition, due to boolean standardization, `enable_snat` 's accepted value is changed from 'yes'/'no' to true/ false. `vpn_access` here did not get renamed, but the accepted value has also changed to boolean.

The updated file should now look something like:

```
# Updated VPN_setup.tf
# ...

resource "aviatrix_gateway" "aws_vpn_gw" {
  cloud_type    = 1
  account_name  = "devops"
  gw_name       = "aws_vpn_gw"
  vpc_id        = "vpc-abc123"
  vpc_reg       = "us-east-1"
  gw_size       = "t2.micro" # vpc_size -> gw_size
  subnet        = "10.0.0.0/24" # vpc_net -> subnet

  # note that dns_server is removed
  vpn_access    = true # "yes" -> true
  vpn_cidr      = "192.168.43.0/24"
  max_vpn_conn  = 100 # newly added
  enable_elb    = "yes"
  elb_name      = "example-elb-name"

  enable_snat   = true # enable_nat = "yes" -> enable_snat = true
  # ...

}
```

Once again, as in the previous two phases, you will have to update your Aviatrix Terraform provider repository and use the correct corresponding release.

Navigate to your local Aviatrix Terraform provider repository, which by default, if setup according to our initial setup doc here (https://github.com/terraform-providers/terraform-provider-aviatrix/blob/master/README.md), is `$GOPATH/src/github.com/terraform-providers` . We will then use Git to pull latest changes from our remote repository ( `git pull` ), and switch to the branch that corresponds with our Controller version ( `git checkout UserConnect-4.7-TF.12-v2` ). Build the provider as according to Step 3 depending on your OS. Finally, navigate back to where your Terraform files reside, and perform a `terraform init` to reinitialise the Terraform environment for the current directory based on the new provider.

Now, to determine whether or not to do a `terraform refresh` or `terraform import` , refer to the rules in Step 3 of this phase. In this example, according to the stated rules, because attributes `vpc_size` and `vpc_net` got renamed, and `enable_nat` got renamed and changed to accept boolean, it would seem this gateway resource would only need a `terraform refresh` . However, because this vpn-gateway uses the attribute `vpn_access` , whose only change is that the accepted value got changed to boolean, this resource would need to a `terraform import` instead, because the import rule would take precedence over refresh. After performing the necessary `terraform refresh / terraform import` (per resource), a `terraform plan` may be performed to catch any deltas. If there are still any deltas, you may fix and repeat again.

---

## Beyond Phase 3: to infinity and beyond ~

- Any updates/ future releases for the Aviatrix Terraform provider will continue to be documented here:
  - Aviatrix Terraform Provider Release Notes (https://www.terraform.io/docs/providers/aviatrix/guides/release-notes.html)
- Any updates/ future releases for R2.0+ that might impact customers will continue to be documented here:

- Aviatrix Terraform Provider R2.x Feature Changelist (https://www.terraform.io/docs/providers/aviatrix/guides/feature-changelist-v2.html)
- Any future necessary changes will only be simple and only require small tweaks and a `terraform refresh`

# aviatrix_account

The aviatrix_account resource allows the creation and management of Aviatrix cloud accounts.

## Example Usage

```
# Create an Aviatrix AWS Account with IAM roles
resource "aviatrix_account" "tempacc" {
  account_name       = "username"
  cloud_type         = 1
  aws_account_number = "123456789012"
  aws_iam            = true
  aws_role_app       = "arn:aws:iam::123456789012:role/aviatrix-role-app"
  aws_role_ec2       = "arn:aws:iam::123456789012:role/aviatrix-role-ec2"
}

# Or you can create an Aviatrix AWS Account with access_key/secret key
resource "aviatrix_account" "tempacc" {
  account_name       = "username"
  cloud_type         = 1
  aws_iam            = false
  aws_account_number = "123456789012"
  aws_access_key     = "ABCDEFGHIJKL"
  aws_secret_key     = "ABCDEFGHIJKLabcdefghijkl"
}

# Create an Aviatrix GCP Account
resource "aviatrix_account" "tempacc_gcp" {
  account_name                       = "username"
  cloud_type                         = 4
  gcloud_project_id                  = "aviatrix-123456"
  gcloud_project_credentials_filepath = "/home/ubuntu/test_gcp/aviatrix-abc123.json"
}

# Create an Aviatrix Azure ARM Account
resource "aviatrix_account" "tempacc_arm" {
  account_name        = "username"
  cloud_type          = 8
  arm_subscription_id = "12345678-abcd-efgh-ijkl-123456789abc"
  arm_directory_id    = "abcdefgh-1234-5678-9100-abc123456789"
  arm_application_id  = "1234abcd-12ab-34cd-56ef-abcdef123456"
  arm_application_key = "213df1SDF1231Gsaf/fa23-4A/324j12390801+FSwe="
}

# Create an Aviatrix Oracle OCI Account
resource "aviatrix_account" "tempacc_oci" {
  account_name                 = "username"
  cloud_type                   = 16
  oci_tenancy_id               = "ocid1.tenancy.oc1..aaaaaaaa"
  oci_user_id                  = "ocid1.user.oc1..aaaaaaaazly"
  oci_compartment_id           = "ocid1.tenancy.oc1..aaaaaaaaxo"
  oci_api_private_key_filepath = "/Users/public/Documents/oci_api_key.pem"
}
```

# Argument Reference

The following arguments are supported:

- `account_name` - (Required) Account name. This can be used for logging in to CloudN console or UserConnect controller.
- `cloud_type` - (Required) Type of cloud service provider. Only AWS, GCP, ARM and OCI are supported currently. Enter 1 for AWS, 4 for GCP, 8 for ARM, 16 for OCI.
- `aws_account_number` - (Optional) AWS Account number to associate with Aviatrix account. Required when creating an account for AWS.
- `aws_iam` - (Optional) AWS IAM-role based flag, this option is for UserConnect.
- `aws_access_key` - (Optional) AWS Access Key. Required when aws_iam is "false" and when creating an account for AWS.
- `aws_secret_key` - (Optional) AWS Secret Key. Required when aws_iam is "false" and when creating an account for AWS.
- `aws_role_app` - (Optional) AWS App role ARN, this option is for UserConnect. Required when aws_iam is "true" and when creating an account for AWS.
- `aws_role_ec2` - (Optional) AWS EC2 role ARN, this option is for UserConnect. Required when aws_iam is "true" and when creating an account for AWS.
- `gcloud_project_id` - (Optional) GCloud Project ID.
- `gcloud_project_credentials_filepath` - (Optional) GCloud Project Credentials [local filepath].json. Required when creating an account for GCP.
- `arm_subscription_id` - (Optional) Azure ARM Subscription ID. Required when creating an account for ARM.
- `arm_directory_id` - (Optional) Azure ARM Directory ID. Required when creating an account for ARM.
- `arm_application_id` - (Optional) Azure ARM Application ID. Required when creating an account for ARM.
- `arm_application_key` - (Optional) Azure ARM Application key. Required when creating an account for ARM.
- `oci_tenancy_id` - (Optional)Oracle OCI Tenancy ID. Required when creating an account for OCI.
- `oci_user_id` - (Optional) Oracle OCI User ID. Required when creating an account for OCI.
- `oci_compartment_id` - (Optional) Oracle OCI Compartment ID. Required when creating an account for OCI.
- `oci_api_private_key_filepath` - (Optional) Oracle OCI API Private Key local file path. Required when creating an account for OCI.

**NOTE:** Please make sure that the IAM roles/profiles have already been created before running this, if aws_iam = "true". More information on the IAM roles is at https://docs.aviatrix.com/HowTos/iam_policies.html (https://docs.aviatrix.com/HowTos/iam_policies.html) and https://docs.aviatrix.com/HowTos/HowTo_IAM_role.html (https://docs.aviatrix.com/HowTos/HowTo_IAM_role.html)

# Import

Instance account can be imported using the account_name (when doing import, needs to leave aws_secret_key blank), e.g.

```
$ terraform import aviatrix_account.test account_name
```

# aviatrix_account_user

The aviatrix_account_user resource allows the creation and management of Aviatrix User Accounts.

## Example Usage

```
# Create an Aviatrix User Account
resource "aviatrix_account_user" "test_accountuser" {
  username     = "username1"
  account_name = "test-accountname"
  email        = "username1@testdomain.com"
  password     = "passwordforuser1-1234"
}
```

## Argument Reference

The following arguments are supported for creating user account:

- `username` - (Required) Name of account user to be created.
- `account_name` - (Required) Cloud account name of user to be created.
- `email` - (Required) Email of address of account user to be created.
- `password` - (Required) Login password for the account user to be created. If password is changed, current account will be destroyed and a new account will be created.

## Import

Instance account_user can be imported using the username (when doing import, needs to leave password argument blank), e.g.

```
$ terraform import aviatrix_account_user.test username
```

# aviatrix_arm_peer

The aviatrix_arm_peer resource allows the creation and management of Aviatrix ARM Peerings.

## Example Usage

```
# Create an Aviatrix ARM Peering
resource "aviatrix_arm_peer" "test_armpeer" {
  account_name1             = "test1-account"
  account_name2             = "test2-account"
  vnet_name_resource_group1 = "vpc-abcd1234"
  vnet_name_resource_group2 = "vpc-rdef3333"
  vnet_reg1                 = "us-east-1"
  vnet_reg2                 = "us-west-1"
}
```

## Argument Reference

The following arguments are supported:

- `account_name1` - (Required) This parameter represents the name of an Azure Cloud-Account in Aviatrix controller.
- `account_name2` - (Required) This parameter represents the name of an Azure Cloud-Account in Aviatrix controller.
- `vnet_name_resource_group1` - (Required) VNet-Name of Azure cloud. Example: "VNet_Name:Resource_Group_Name".
- `vnet_name_resource_group2` - (Required) VNet-Name of Azure cloud. Example: "VNet_Name:Resource_Group_Name".
- `vnet_reg1` - (Required) Region of Azure cloud. Example: "East US 2".
- `vnet_reg2` - (Required) Region of Azure cloud. Example: "East US 2".

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `vnet_cidr1` - List of VNet CIDR of vnet_name_resource_group1.
- `vnet_cidr2` - List of VNet CIDR of vnet_name_resource_group2.

## Import

Instance arm_peer can be imported using the vnet_name_resource_group1 and vnet_name_resource_group2, e.g.

```
$ terraform import aviatrix_aws_peer.test vnet_name_resource_group1~vnet_name_resource_group2
```

# aviatrix_aws_peer

The aviatrix_aws_peer resource allows the creation and management of Aviatrix AWS Peerings.

## Example Usage

```
# Create an Aviatrix AWS Peering
resource "aviatrix_aws_peer" "test_awspeer" {
  account_name1 = "test1-account"
  account_name2 = "test2-account"
  vpc_id1       = "vpc-abcd1234"
  vpc_id2       = "vpc-rdef3333"
  vpc_reg1      = "us-east-1"
  vpc_reg2      = "us-west-1"
  rtb_list1     = [
    "rtb-abcd1234",
  ]
  rtb_list2     = [
    "rtb-wxyz5678",
  ]
}
```

## Argument Reference

The following arguments are supported:

- `account_name1` - (Required) This parameter represents the name of an AWS Cloud-Account in Aviatrix controller.
- `account_name2` - (Required) This parameter represents the name of an AWS Cloud-Account in Aviatrix controller.
- `vpc_id1` - (Required) VPC-ID of AWS cloud. Example: AWS: "vpc-abcd1234".
- `vpc_id2` - (Required) VPC-ID of AWS cloud. Example: AWS: "vpc-abcd1234".
- `vpc_reg1` - (Required) Region of AWS cloud. Example: AWS: "us-east-1".
- `vpc_reg2` - (Required) Region of AWS cloud. Example: AWS: "us-east-1".
- `rtb_list1` - (Optional) List of Route table ID. Valid Values: ["all"], ["rtb-abcd1234"] OR ["rtb-abcd1234,rtb-wxyz5678"].
- `rtb_list2` - (Optional) List of Route table ID. Valid Values: ["all"], ["rtb-abcd1234"] OR ["rtb-abcd1234,rtb-wxyz5678"].

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `rtb_list1_output` - List of route table ID of vpc_id1.
- `rtb_list2_output` - List of route table ID of vpc_id2.

# Import

Instance aws_peer can be imported using the vpc_id1 and vpc_id2, e.g.

```
$ terraform import aviatrix_aws_peer.test vpc_id1~vpc_id2
```

# aviatrix_aws_tgw

The aviatrix_aws_tgw resource allows the creation and management of AWS TGWs.

## Example Usage

```
# Create an Aviatrix AWS TGW
resource "aviatrix_aws_tgw" "test_aws_tgw" {
  account_name                     = "devops"
  attached_aviatrix_transit_gateway = [
     "avxtransitgw",
     "avxtransitgw2"
  ]
  aws_side_as_number               = "64512"
  manage_vpc_attachment            = true
  region                           = "us-east-1"
  tgw_name                         = "testAWSTgw"

  security_domains {
    connected_domains    = [
      "Default_Domain",
      "Shared_Service_Domain",
      "mysdn1"
    ]
    security_domain_name = "Aviatrix_Edge_Domain"
  }

  security_domains {
    connected_domains    = [
      "Aviatrix_Edge_Domain",
      "Shared_Service_Domain"
    ]
    security_domain_name = "Default_Domain"
  }

  security_domains {
    connected_domains    = [
      "Aviatrix_Edge_Domain",
      "Default_Domain"
    ]
    security_domain_name = "Shared_Service_Domain"
  }

  security_domains {
    connected_domains    = [
      "Aviatrix_Edge_Domain"
    ]
    security_domain_name = "SDN1"

    attached_vpc {
      vpc_account_name = "devops1"
      vpc_id           = "vpc-0e2fac2b91"
      vpc_region       = "us-east-1"
    }
```

```
    attached_vpc {
      vpc_account_name = "devops1"
      vpc_id           = "vpc-0c63660a16"
      vpc_region       = "us-east-1"
    }

    attached_vpc {
      vpc_account_name = "devops"
      vpc_id           = "vpc-032005cc371"
      vpc_region       = "us-east-1"
    }
  }

  security_domains {
    security_domain_name = "mysdn2"

    attached_vpc {
      vpc_region       = "us-east-1"
      vpc_account_name = "devops"
      vpc_id           = "vpc-032005cc371"
    }
  }
}
```

## Argument Reference

The following arguments are supported:

- `tgw_name` - (Required) Name of the AWS TGW which is going to be created.
- `account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `region` - (Required) Region of cloud provider(AWS).
- `aws_side_as_number` - (Required) BGP Local ASN (Autonomous System Number). Integer between 1-65535. Example: "65001".
- `security_domains` - (Required) Security Domains to create together with AWS TGW's creation. Three default domains are created automatically together with the AWS TGW's creation, so are the connections between any two of them. These three domains can't be deleted, but the connection between any two of them can be deleted.
  - `security_domain_name` - (Required) Three default domains ("Aviatrix_Edge_Domain", "Default_Domain" and "Shared_Service_Domain") are required with AWS TGW's creation.
  - `aviatrix_firewall` - (Optional) Set to true if the security domain is an aviatrix firewall domain. Valid values: true, false. Default value: false.
  - `native_egress` - (Optional) Set to true if the security domain is a native egress domain. Valid values: true, false. Default value: false.
  - `native_firewall` - (Optional) Set to true if the security domain is a native firewall domain. Valid values: true, false. Default value: false.
  - `connected_domains` - (Optional) A list of domains connected to the domain (name: `security_domain_name` ) together with its creation.

- `attached_vpc` - (Optional) A list of VPCs attached to the domain (name: `security_domain_name`) together with its creation. This list needs to be null for "Aviatrix_Edge_Domain".
- `vpc_region` - (Required) Region of the vpc, needs to be consistent with AWS TGW's region.
- `vpc_account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `vpc_id` - (Required) This parameter represents the ID of the VPC which is going to be attached to the security domain (name: `security_domain_name`) which is going to be created.

- `attached_aviatrix_transit_gateway` - (Optional) A list of Names of Aviatrix Transit Gateway to attach to one of the three default domains: Aviatrix_Edge_Domain.
- `manage_vpc_attachment` - (Optional) This parameter is a switch used to allow attaching VPCs to tgw using the aviatrix_aws_tgw resource. If it is set to false, attachment of vpc must be done using the aviatrix_aws_tgw_vpc_attachment resource. Valid values: true or false. Default value is true.

**NOTE:** `manage_vpc_attachment` - If you are using/upgraded to Aviatrix Terraform Provider R1.5+, and an aws_tgw resource was originally created with a provider version <R1.5, you must do 'terraform refresh' to update and apply the attribute's default value (true) into the state file.

## Import

Instance aws_tgw can be imported using the tgw_name, e.g.

```
$ terraform import aviatrix_aws_tgw.test tgw_name
```

**NOTE:** If "manage_vpc_attachment" is set to "false", import action will also import the information of the VPCs attached to tgw into the state file. Will need to do "Terraform Apply" to sync "manage_vpc_attachment" to "false".

# aviatrix_aws_tgw

The aviatrix_aws_tgw resource manages attaching or detaching VPCs to/from an AWS TGW.

## Example Usage

```
# Create an Aviatrix AWS TGW VPC Attachment
resource "aviatrix_aws_tgw_vpc_attachment" "test_aws_tgw_vpc_attachment" {
  tgw_name             = "tgwTest"
  region               = "us-east-1"
  security_domain_name = "mySdn"
  vpc_account_name     = "accountTest"
  vpc_id               = "vpc-0e2fac2b91c6697b3"
}
```

## Argument Reference

The following arguments are supported:

- `tgw_name` - (Required) Name of the AWS TGW.
- `region` - (Required) Region of cloud provider(AWS).
- `security_domain_name` - (Required & ForceNew) The name of the security domain, to which the VPC will be attached. If changed, the VPC will be detached from the old domain, and attached to the new domain.
- `vpc_account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller, which is associated with the VPC.
- `vpc_id` - (Required) This parameter represents the ID of the VPC which is going to be attached to the security domain (name: `security_domain_name` ).

## Import

Instance aws_tgw_vpc_attachment can be imported using the tgw_name, security_domain_name and vpc_id, e.g.

```
$ terraform import aviatrix_aws_tgw_vpc_attachment.test tgw_name~security_domain_name~vpc_id
```

# aviatrix_aws_tgw_vpn_conn

The aviatrix_aws_tgw_vpn_conn resource allows the creation and management of Aviatrix AWS TGW VPN Connections.

## Example Usage

```
# Create an Aviatrix AWS TGW VPN Connection (dynamic)
resource "aviatrix_aws_tgw_vpn_conn" "test_aws_tgw_vpn_conn" {
  tgw_name          = "myawstgw1"
  route_domain_name = "Default_Domain"
  connection_name   = "myConn1"
  public_ip         = "40.0.0.0"
  remote_as_number  = "12"
}

# Create an Aviatrix AWS TGW VPN Connection (static)
resource "aviatrix_aws_tgw_vpn_conn" "test_aws_tgw_vpn_conn" {
  tgw_name          = "myawstgw1"
  route_domain_name = "Default_Domain"
  connection_name   = "myConn1"
  public_ip         = "40.0.0.0"
  remote_cidr       = "16.0.0.0/16,16.1.0.0/16"
}
```

## Argument Reference

The following arguments are supported:

- `tgw_name` - (Required) This parameter represents the name of an AWS TGW.
- `route_domain_name` - (Required) The name of a route domain, to which the vpn will be attached. Only "Default_Domain" is supported now.
- `connection_name` - (Required) Unique name of the connection.
- `public_ip` - (Required) Public IP address. Example: "40.0.0.0".
- `remote_as_number` - (Optional) AWS side as a number. Integer between 1-65535. Example: "12".
- `remote_cidr` - (Optional) Remote CIDRs separated by ",". Example: AWS: "16.0.0.0/16,16.1.0.0/16".
- `inside_ip_cidr_tun_1` - (Optional) Inside IP CIDR for Tunnel 1. A /30 CIDR in 169.254.0.0/16.
- `pre_shared_key_tun_1` - (Optional) Pre-Shared Key for Tunnel 1. A 8-64 character string with alphanumeric underscore(_) and dot(.). It cannot start with 0.
- `inside_ip_cidr_tun_2` - (Optional) Inside IP CIDR for Tunnel 2. A /30 CIDR in 169.254.0.0/16.
- `pre_shared_key_tun_2` - (Optional) Pre-Shared Key for Tunnel 2. A 8-64 character string with alphanumeric underscore(_) and dot(.). It cannot start with 0.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `vpn_id` - ID of the vpn generated by creation of the connection.

## Import

Instance aws_tgw_vpn_conn can be imported using the tgw_name and vpn_id, e.g.

```
$ terraform import aviatrix_aws_tgw_vpn_conn.test tgw_name~vpn_id
```

# aviatrix_controller_config

The aviatrix_controller_config resource allows the creation and management of an Aviatrix controller config.

## Example Usage

```
# Create an Aviatrix Controller Config
resource "aviatrix_controller_config" "test_controller_config" {
  sg_management_account_name = "username"
  http_access                = true
  fqdn_exception_rule        = false
  security_group_management  = true
}
```

```
# Create an Aviatrix Controller Config with Controller Upgrade
resource "aviatrix_controller_config" "test_controller_config" {
  sg_management_account_name = "username"
  http_access                = true
  fqdn_exception_rule        = false
  security_group_management  = true
  target_version             = "latest"
}
```

## Argument Reference

The following arguments are supported:

- `sg_management_account_name` - (Optional) Cloud account name of user.
- `http_access` - (Optional) Switch for http access. Valid values: true, false. Default value: false.
- `fqdn_exception_rule` - (Optional) A system-wide mode. Valida values: true, false. Defaultvalue: true.
- `security_group_management` - (Optional) Used to manage the Controller instance's inbound rules from gateways. Valid values: true, false. Default value: false.
- `target_version` - (Optional) The release version number to which the controller will be upgraded to. If not specified, controller will not be upgraded. If set to "latest", controller will be upgraded to the latest release. Please look at https://docs.aviatrix.com/HowTos/inline_upgrade.html (https://docs.aviatrix.com/HowTos/inline_upgrade.html) for more information.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `version` - Current version of the controller.

# Import

Instance controller_config can be imported using controller IP, e.g. controller IP is : 10.11.12.13

```
$ terraform import aviatrix_controller_config.test 10-11-12-13
```

# aviatrix_firenet

The aviatrix_firenet resource allows the creation and management of Aviatrix FireNets.

## Example Usage

```
# Create an Aviatrix FireNet associated to a Firewall Instance
resource "aviatrix_firenet" "test_firenet" {
  vpc_id             = "vpc-032005cc371"
  inspection_enabled = true
  egress_enabled     = false

  firewall_instance_association {
    firenet_gw_name      = "avx_firenet_gw"
    instance_id          = "i-09dc118db6a1eb901"
    firewall_name        = "avx_firewall_instance"
    attached             = true
    lan_interface        = "eni-0a34b1827bf222353"
    management_interface = "eni-030e53176c7f7d34a"
    egress_interface     = "eni-03b8dd53a1a731481"
  }
}

# Create an Aviatrix FireNet associated to an FQDN Gateway
resource "aviatrix_firenet" "test_firenet" {
  vpc_id             = "vpc-032005cc371"
  inspection_enabled = true
  egress_enabled     = false

  firewall_instance_association {
    gw_name       = "avx_firenet_gw"
    instance_id   = "avx_fqdn_gateway"
    vendor_type   = "fqdn_gateway"
    attached      = true
  }
}
```

## Argument Reference

The following arguments are supported:

- `vpc_id` - (Required) ID of the Security VPC.
- `inspection_enabled` - (Optional) Enable/Disable traffic inspection. Valid values: true, false. Default value: true.
- `egress_enabled` - (Optional) Enable/Disable egress through firewall. Valid values: true, false. Default value: false.
- `firewall_instance_association` - (Optional) List of firewall instances to be associated with fireNet.
  - `firenet_gw_name` - (Required) Name of the primary FireNet gateway.
  - `vendor_type` - (Required) Type of the firewall. Valid values: "Generic", "fqdn_gateway". Default value: "Generic".

- `instance_id` - (Required) ID of Firewall instance, if associating FQDN gateway to fireNet, it is FQDN gateway's gw_name..
- `firewall_name` - (Optional) Firewall instance name, required if it is a firewall instance.
- `lan_interface` - (Optional) Lan interface ID, required if it is a firewall instance.
- `management_interface` - (Optional) Management interface ID, required if it is a firewall instance.
- `egress_interface` - (Optional) Egress interface ID, required if it is a firewall instance.
- `attached` - (Optional) Switch to attach/detach firewall instance to/from fireNet. Valid values: true, false. Default value: false.

**NOTE:** `inspection_enabled` - Default value is true for associating firewall instance to fireNet. Only false is supported for associating FQDN gateway to fireNet.

**NOTE:** `egress_enabled` - Default value is false for associating firewall instance to fireNet. Only true is supported for associating FQDN gateway to fireNet.

**NOTE:** `firewall_instance_association` - If associating FQDN gateway to fireNet, "single_az_ha" needs to be enabled for the FQDN gateway.

## Import

Instance firenet can be imported using the vpc_id, e.g.

```
$ terraform import aviatrix_firenet.test vpc_id
```

# aviatrix_firewall

The aviatrix_firewall resource allows the creation and management of Aviatrix Firewall Policies.

## Example Usage

```
# Create an Aviatrix Firewall
resource "aviatrix_firewall" "test_firewall" {
  gw_name          = "gateway-1"
  base_policy      = "allow-all"
  base_log_enabled = true

  policy {
    protocol    = "tcp"
    src_ip      = "10.15.0.224/32"
    log_enabled = false
    dst_ip      = "10.12.0.172/32"
    action      = "deny"
    port        = "0:65535"
    description = "This is policy no.1"
  }

  policy {
    protocol    = "tcp"
    src_ip      = "10.15.1.224/32"
    log_enabled = false
    dst_ip      = "10.12.1.172/32"
    action      = "deny"
    port        = "0:65535"
    description = "This is policy no.2"
  }
}
```

## Argument Reference

The following arguments are supported:

- `gw_name` - (Required) The name of gateway.
- `base_policy` - (Optional) New base policy. Valid Values: "allow-all", "deny-all".
- `base_log_enabled` - (Optional) Indicates whether enable logging or not. Valid Values: true, false.
- `policy` - (Optional) New access policy for the gateway. Type: String (valid JSON). Seven fields are required for each policy item: src_ip, dst_ip, protocol, port, allow_deny, log_enabled and description.
  - `src_ip` - (Required) CIDRs separated by comma or tag names such "HR" or "marketing" etc. Example: "10.30.0.0/16,10.45.0.0/20". The aviatrix_firewall_tag resource should be created prior to using the tag name.
  - `dst_ip` - (Required) CIDRs separated by comma or tag names such "HR" or "marketing" etc. Example: "10.30.0.0/16,10.45.0.0/20". The aviatrix_firewall_tag resource should be created prior to using the tag name.

- `protocol` - (Optional): "all", "tcp", "udp", "icmp", "sctp", "rdp", "dccp".
- `port` - (Required) a single port or a range of port numbers. Example: "25", "25:1024".
- `action` - (Required) Valid values: "allow", "deny".
- `log_enabled` - (Optional) Valid values: true, false. Default value: false.
- `description` - (Optional) Description of the policy. Example: "This is policy no.1".

## Import

Instance firewall can be imported using the gw_name, e.g.

```
$ terraform import aviatrix_firewall.test gw_name
```

# aviatrix_firewall_instance

The aviatrix_firewall_instance resource allows the creation and deletion of Aviatrix Firewall Instances.

## Example Usage

```
# Create an Aviatrix Firewall Instance
resource "aviatrix_firewall_instance" "test_firewall_instance" {
  vpc_id            = "vpc-032005cc371"
  firenet_gw_name   = "avx_firenet_gw"
  firewall_name     = "avx_firewall_instance"
  firewall_image    = "Palo Alto Networks VM-Series Next-Generation Firewall Bundle 1"
  firewall_size     = "m5.xlarge"
  management_subnet = "10.4.0.16/28"
  egress_subnet     = "10.4.0.32/28"
}
```

## Argument Reference

The following arguments are supported:

- `vpc_id` - (Required) ID of the Security VPC.
- `firenet_gw_name` - (Required) Name of the primary FireNet gateway.
- `firewall_name` - (Required) Name of the firewall instance to be created.
- `firewall_image` - (Required) One of the AWS AMIs from Palo Alto Networks.
- `firewall_size` - (Required) Instance size of the firewall. Example: "m5.xlarge".
- `management_subnet` - (Required) Management Interface Subnet. Select the subnet whose name contains “gateway and firewall management”.
- `egress_subnet` - (Required) Egress Interface Subnet. Select the subnet whose name contains “FW-ingress-egress”.
- `key_name` - (Optional) The .pem file name for SSH access to the firewall instance.
- `iam_role` - (Optional) In advanced mode, create an IAM Role on the AWS account that launched the FireNet gateway. Create a policy to attach to the role. The policy is to allow access to “Bootstrap Bucket”.
- `bootstrap_bucket_name` - (Optional) In advanced mode, specify a bootstrap bucket name where the initial configuration and policy file is stored.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `instance_id` - ID of the firewall instance created.

- `lan_interface` - ID of Lan Interface created.
- `management_interface` - ID of Management Interface created.
- `egress_interface` - ID of Egress Interface created.
- `public_ip` - Management Public IP.

## Import

Instance firewall_instance can be imported using the instance_id, e.g.

```
$ terraform import aviatrix_firewall_instance.test instance_id
```

# aviatrix_firewall_tag

The aviatrix_firewall_tag resource allows the creation and management of Aviatrix Firewall Tags.

## Example Usage

```
# Create an Aviatrix Firewall Tag
resource "aviatrix_firewall_tag" "test_firewall_tag" {
  firewall_tag = "test-firewall-tag"

  cidr_list {
    cidr_tag_name = "a1"
    cidr          = "10.1.0.0/24"
  }

  cidr_list {
    cidr_tag_name = "b1"
    cidr          = "10.2.0.0/24"
  }
}
```

## Argument Reference

The following arguments are supported:

- `firewall_tag` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `cidr_list` - (Optional) A JSON file with the following:
  - `cidr_tag_name` - (Required) The name attribute of a policy. Example: "policy1".
  - `cidr` - (Required) The CIDR attribute of a policy. Example: "10.88.88.88/32".

## Import

Instance firewall_tag can be imported using the firewall_tag, e.g.

```
$ terraform import aviatrix_firewall_tag.test firewall_tag
```

# aviatrix_fqdn

The aviatrix_fqdn resource manages FQDN filtering for Aviatrix Gateway.

## Example Usage

```
# Create an Aviatrix Gateway FQDN filter
resource "aviatrix_fqdn" "test_fqdn" {
  fqdn_tag     = "my_tag"
  fqdn_enabled = true
  fqdn_mode    = "white"

  gw_filter_tag_list {
    gw_name        = "gwTest1"
    source_ip_list = [
      "172.31.0.0/16",
      "172.31.0.0/20"
    ]
  }

  gw_filter_tag_list {
    gw_name        = "gwTest2"
    source_ip_list = [
      "30.0.0.0/16"
    ]
  }

  domain_names {
    fqdn  = "facebook.com"
    proto = "tcp"
    port  = "443"
  }

  domain_names {
    fqdn  = "reddit.com"
    proto = "tcp"
    port  = "443"
  }
}
```

## Argument Reference

The following arguments are supported:

- `fqdn_tag` - (Required) FQDN Filter Tag Name.
- `fqdn_enabled` - (Optional) FQDN Filter Tag Status. Valid values: true, false.
- `fqdn_mode` - (Optional) Specify the tag color to be a white-list tag or black-list tag. Valid values: "white", "black".
- `gw_filter_tag_list` - (Optional) A list of gateways to attach to the specific tag.

    - `gw_name` - (Required) Name of the gateway to attach to the specific tag.
    - `source_ip_list` - (Optional) List of source IPs in the VPC qualified for a specific tag.
- `domain_names` - (Optional) One or more domain names in a list with details as listed below:
    - `fqdn` - (Required) FQDN. Example: "facebook.com".
    - `proto` - (Required) Protocol. Valid values: "all", "tcp", "udp", "icmp".
    - `port` - (Required) Port. Example "25".
    - For protocol "all", port must be set to "all".
    - For protocol “icmp”, port must be set to “ping”.

**NOTE:** If you are using/upgraded to Aviatrix Terraform Provider R1.5+, and an fqdn resource was originally created with a provider version <R1.5, you must modify your configuration file to match current format, and do ‘terraform refresh’ to update the state file to current format.

**NOTE:** In order for the FQDN feature to be enabled, “enable_snat” must be set to true in the specified gateway. If it is not set at gateway creation, creation of FQDN resource will automatically enable SNAT and users must rectify the diff in the Terraform state by setting "enable_nat = true"in their gateway resource.

## Import

Instance fqdn can be imported using the fqdn_tag, e.g.

```
$ terraform import aviatrix_fqdn.test fqdn_tag
```

# aviatrix_gateway

The aviatrix_gateway resource allows the creation and management of Aviatrix gateways.

## Example Usage

```
# Create an Aviatrix AWS Gateway
resource "aviatrix_gateway" "test_gateway_aws" {
  cloud_type   = 1
  account_name = "devops"
  gw_name      = "avtxgw1"
  vpc_id       = "vpc-abcdef"
  vpc_reg      = "us-west-1"
  gw_size      = "t2.micro"
  subnet       = "10.0.0.0/24"
  tag_list     = [
    "k1:v1",
    "k2:v2",
  ]
}

# Create an Aviatrix AWS Gateway with VPN enabled
resource "aviatrix_gateway" "test_gateway_aws" {
  cloud_type   = 1
  account_name = "devops"
  gw_name      = "avtxgw1"
  vpc_id       = "vpc-abcdef"
  vpc_reg      = "us-west-1"
  gw_size      = "t2.micro"
  subnet       = "10.0.0.0/24"
  vpn_access   = "yes"
  vpn_cidr     = "192.168.43.0/24"
  max_vpn_conn = "100"
}

# Create an Aviatrix GCP Gateway
resource "aviatrix_gateway" "test_gateway_gcp" {
  cloud_type   = 4
  account_name = "devops-gcp"
  gw_name      = "avtxgw-gcp"
  vpc_id       = "gcp-gw-vpc"
  vpc_reg      = "us-west1-b"
  gw_size      = "n1-standard-1"
  subnet       = "10.12.0.0/24"
}

# Create an Aviatrix ARM Gateway
resource "aviatrix_gateway" "test_gateway_arm" {
  cloud_type   = 8
  account_name = "devops-arm"
  gw_name      = "avtxgw-arm"
  vpc_id       = "gateway:test-gw-123"
  vpc_reg      = "West US"
  gw_size      = "Standard_D2"
  subnet       = "10.13.0.0/24"
```

```
}

# Create an Aviatrix Oracle Gateway
resource "aviatrix_gateway" "test_gateway_oracle" {
  cloud_type   = 16
  account_name = "devops-oracle"
  gw_name      = "avtxgw-oracle"
  vpc_id       = "vpc-oracle-test"
  vpc_reg      = "us-ashburn-1"
  gw_size      = "VM.Standard2.2"
  subnet       = "10.7.0.0/16"
}

# Create an Aviatrix AWS Gateway with Peering HA enabled
resource "aviatrix_gateway" "test_gateway_aws" {
  cloud_type         = 1
  account_name       = "devops"
  gw_name            = "avtxgw1"
  vpc_id             = "vpc-abcdef"
  vpc_reg            = "us-west-1"
  gw_size            = "t2.micro"
  subnet             = "10.0.0.0/24"
  peering_ha_subnet  = "10.0.0.0/24"
  peering_ha_gw_size = "t2.micro"
}

# Create an Aviatrix GCP Gateway with Peering HA enabled
resource "aviatrix_gateway" "test_gateway_gcp" {
  cloud_type         = 4
  account_name       = "devops-gcp"
  gw_name            = "avtxgw-gcp"
  vpc_id             = "gcp-gw-vpc"
  vpc_reg            = "us-west1-b"
  gw_size            = "n1-standard-1"
  subnet             = "10.12.0.0/24"
  peering_ha_zone    = "us-west1-c"
  peering_ha_gw_size = "n1-standard-1"
}
```

## Argument Reference

The following arguments are supported:

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1), GCP(4), ARM(8), and OCI(16) are supported.
- `account_name` - (Required) Account name. This account will be used to launch Aviatrix gateway.
- `gw_name` - (Required) Aviatrix gateway unique name.
- `vpc_id` - (Required) VPC-ID/VNet-Name of cloud provider. Required if for aws. Example: AWS: "vpc-abcd1234", GCP: "vpc-gcp-test", ARM: "vnet1:hello", OCI: "vpc-oracle-test1".
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", GCP: "us-west2-a", ARM: "East US 2", Oracle: "us-ashburn-1".

- `gw_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", ARM: "Standard_B1s", Oracle: "VM.Standard2.2", GCP: "n1-standard-1".
- `subnet` - (Required) A VPC Network address range selected from one of the available network ranges. Example: "172.31.0.0/20".
- `insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Gateway. Required for AWS if insane_mode is set. Example: AWS: "us-west-1a".
- `enable_snat` - (Optional) Enable Source NAT for this container. Supported values: true, false. Default value: false.
- `vpn_access` - (Optional) Enable user access through VPN to this container. Supported values: true, false.
- `vpn_cidr` - (Optional) VPN CIDR block for the container. Required if vpn_access is true. Example: "192.168.43.0/24".
- `max_vpn_conn` - (Optional) Maximum number of active VPN users allowed to be connected to this gateway. Required if vpn_access is true. Make sure the number is smaller than the VPN CIDR block. Example: 100.
- `enable_elb` - (Optional) Specify whether to enable ELB or not. Not supported for Oracle gateways. Supported values: true, false.
- `elb_name` - (Optional) A name for the ELB that is created. If it is not specified, a name is generated automatically.
- `split_tunnel` - (Optional) Specify split tunnel mode. Supported values: true, false.
- `name_servers` - (Optional) A list of DNS servers used to resolve domain names by a connected VPN user when Split Tunnel Mode is enabled.
- `search_domains` - (Optional) A list of domain names that will use the NameServer when a specific name is not in the destination when Split Tunnel Mode is enabled.
- `additional_cidrs` - (Optional) A list of destination CIDR ranges that will also go through the VPN tunnel when Split Tunnel Mode is enabled.
- `otp_mode` - (Optional) Two step authentication mode. "2": DUO, "3": Okta.
- `saml_enabled` - (Optional) This field indicates whether enabling SAML or not. This field is available in controller version 3.3 or later release. Supported values: true, false.
- `okta_token` - (Optional) Token for Okta auth mode. Required if otp_mode is "3".
- `okta_url` - (Optional) URL for Okta auth mode. Required if otp_mode is "3".
- `okta_username_suffix` - (Optional) Username suffix for Okta auth mode. Example: "aviatrix.com".
- `duo_integration_key` - (Optional) Integration key for DUO auth mode. Required if otp_mode is "2".
- `duo_secret_key` - (Optional) Secret key for DUO auth mode. Required if otp_mode is "2".
- `duo_api_hostname` - (Optional) API hostname for DUO auth mode. Required: Yes if otp_mode is "2".
- `duo_push_mode` - (Optional) Push mode for DUO auth. Required if otp_mode is "2". Valid values: "auto", "selective" and "token".
- `enable_ldap` - (Optional) Specify whether to enable LDAP or not. Supported values: true, false.
- `ldap_server` - (Optional) LDAP server address. Required if enable_ldap is true.
- `ldap_bind_dn` - (Optional) LDAP bind DN. Required if enable_ldap is true.

- `ldap_password` - (Optional) LDAP password. Required if enable_ldap is true.
- `ldap_base_dn` - (Optional) LDAP base DN. Required if enable_ldap is true.
- `ldap_username_attribute` - (Optional) LDAP user attribute. Required if enable_ldap is true.
- `peering_ha_subnet` - (Optional) Public Subnet Information while creating Peering HA Gateway, only subnet is accepted. Required for AWS/ARM if enabling Peering HA. Example: AWS: "10.0.0.0/16".
- `peering_ha_zone` - (Optional) Zone information for creating Peering HA Gateway, only zone is accepted. Required for GCP if enabling Peering HA. Example: GCP: "us-west1-c".
- `peering_ha_insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Peering HA Gateway. Required for AWS if insane_mode is set and peering_ha_subnet is set. Example: AWS: "us-west-1a".
- `peering_ha_eip` - (Optional) Public IP address that you want assigned to the HA peering instance. Only available for AWS.
- `peering_ha_gw_size` - (Optional) Size of the Peering HA Gateway.
- `single_az_ha` (Optional) Set to true if this feature is desired. Supported values: true, false.
- `allocate_new_eip` - (Optional) When value is false, reuse an idle address in Elastic IP pool for this gateway. Otherwise, allocate a new Elastic IP and use it for this gateway. Available in controller 2.7 or later release. Supported values: true, false. Default: true. Option not available for GCP, ARM and Oracle gateways, they will automatically allocate new eip's.
- `eip` - (Optional) Required when allocate_new_eip is false. It uses specified EIP for this gateway. Available in controller 3.5 or later release. Only available for AWS.
- `tag_list` - (Optional) Instance tag of cloud provider. Only available for AWS. Example: ["key1:value1", "key2:value2"].
- `insane_mode` - (Optional) Enable Insane Mode for Gateway. Insane Mode Gateway size must be at least c5 (AWS) or Standard_D3_v2 (ARM). If set, will look for spare /26 segment to create a new subnet. Only supported for AWS or ARM. Supported values: true, false.
- `enable_vpc_dns_server` - (Optional) Enable VPC DNS Server for Gateway. Currently only supports AWS. Valid values: true, false. Default value: false.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `public_ip` - Public IP address of the Gateway created.
- `backup_public_ip` - Private IP address of the Gateway created.
- `public_dns_server` - DNS server used by the gateway. Default is "8.8.8.8", can be overridden with the VPC's setting.
- `security_group_id` - Security group used for the gateway.
- `cloud_instance_id` - Instance ID of the gateway.
- `cloudn_bkup_gateway_inst_id` - Instance ID of the backup gateway.

The following arguments are deprecated:

- `dns_server` - Specify the DNS IP, only required while using a custom private DNS for the VPC.

**NOTE:** `peering_ha_gw_size` - If you are using/upgraded to Aviatrix Terraform Provider R1.8+, and a peering-HA gateway was originally created with a provider version <R1.8, you must do a 'terraform refresh' to update and apply the attribute's value into the state. In addition, you must also input this attribute and its value to its corresponding gateway resource in your `.tf` file.

**NOTE:** `enable_snat` - In order for the FQDN feature to be enabled for the specified gateway, "enable_snat" must be set to true. If it is not set at gateway creation, creation of FQDN resource will automatically enable SNAT and users must rectify the diff in the Terraform state by setting "enable_snat = true" in their config file.

**NOTE:** `max_vpn_conn` - If you are using/upgraded to Aviatrix Terraform Provider R1.14+, and a gateway with VPN enabled was originally created with a provider version <R1.14, you must do a 'terraform refresh' to update and apply the attribute's value into the state. In addition, you must also input this attribute and its value to "100" in your `.tf` file.

## Import

Instance gateway can be imported using the gw_name, e.g.

```
$ terraform import aviatrix_gateway.test gw_name
```

# aviatrix_saml_endpoint

The Account resource allows the creation and management of an Aviatrix SAML Endpoint.

## Example Usage

```
# Create Aviatrix AWS SAML Endpoint
resource "aviatrix_saml_endpoint" "test_saml_endpoint" {
  endpoint_name     = "saml-test"
  idp_metadata_type = "Text"
  idp_metadata      = "${var.idp_metadata}"
}
```

## Argument Reference

The following arguments are supported:

- `endpoint_name` - (Required) The SAML Endpoint name.
- `idp_metadata_type` - (Required) The IDP Metadata type. At the moment only "Text" is supported.
- `idp_metadata` - (Required) The IDP Metadata from SAML provider. Normally the metadata is in XML format which may contain special characters. Best practice is encode metadata in base64 and set here `${base64decode(var.idp_metadata)}`.
- `custom_entity_id` - (Required) Custom Entity ID. Required to be non-empty for 'Custom' Entity ID type, empty for 'Hostname' Entity ID type.

## Import

Instance saml_endpoint can be imported using the SAML Endpoint name, e.g.

```
$ terraform import aviatrix_saml_endpoint.test saml-test
```

# aviatrix_site2cloud

The aviatrix_site2cloud resource Creates and Manages Aviatrix Site2Cloud connections.

## Example Usage

```
# Create an Aviatrix Site2cloud
resource "aviatrix_site2cloud" "test_s2c" {
  vpc_id                     = "vpc-abcd1234"
  connection_name            = "my_conn"
  connection_type            = "unmapped"
  remote_gateway_type        = "generic"
  tunnel_type                = "udp"
  primary_cloud_gateway_name = "gw1"
  remote_gateway_ip          = "5.5.5.5"
  remote_subnet_cidr         = "10.23.0.0/24"
  local_subnet_cidr          = "10.20.1.0/24"
}
```

## Argument Reference

The following arguments are supported:

- `primary_cloud_gateway_name` - (Required) Primary Cloud Gateway Name.
- `backup_gateway_name` - (Optional) Backup gateway name.
- `vpc_id` - (Required) VPC Id of the cloud gateway.
- `connection_name` - (Required) Site2Cloud Connection Name.
- `connection_type` - (Required) Connection Type. Valid Values: "mapped", "unmapped".
- `tunnel_type` - (Required) Site2Cloud Tunnel Type. Valid Values: "udp", "tcp".
- `remote_gateway_type` - (Required) Remote Gateway Type. Valid Values: "generic", "avx", "aws", "azure", "sonicwall", "oracle".
- `remote_gateway_ip` - (Required) Remote Gateway IP.
- `backup_remote_gateway_ip` - (Optional) Backup Remote Gateway IP.
- `pre_shared_key` - (Optional) Pre-Shared Key.
- `backup_pre_shared_key` - (Optional) Backup Pre-Shared Key.
- `remote_subnet_cidr` - (Required) Remote Subnet CIDR.
- `local_subnet_cidr` - (Optional) Local Subnet CIDR. Required for connection type "mapped".
- `remote_subnet_virtual` - Remote Subnet CIDR (Virtual). Required for connection type "mapped" only.

- `local_subnet_virtual` - Local Subnet CIDR (Virtual). Required for connection type "mapped" only.
- `ha_enabled` - (Optional) Specify whether enabling HA or not. Valid Values: true, false.
- `custom_algorithms` - (Optional) Switch to enable custom/non-default algorithms for IPSec Authentication/Encryption. Valid values: true, false.
- `phase_1_authentication` - (Optional) Phase one Authentication. Valid values: 'SHA-1', 'SHA-256', 'SHA-384' and 'SHA-512'. Default value: 'SHA-1'.
- `phase_2_authentication` - (Optional) Phase two Authentication. Valid values: 'NO-AUTH', 'HMAC-SHA-1', 'HMAC-SHA-256', 'HMAC-SHA-384' and 'HMAC-SHA-512'. Default value: 'HMAC-SHA-1'.
- `phase_1_dh_groups` - (Optional) Phase one DH Groups. Valid values: '1', '2', '5', '14', '15', '16', '17' and '18'. Default value: '2'.
- `phase_2_dh_groups` - (Optional) Phase two DH Groups. Valid values: '1', '2', '5', '14', '15', '16', '17' and '18'. Default value: '2'.
- `phase_1_encryption` - (Optional) Phase one Encryption. Valid values: '3DES', 'AES-128-CBC', 'AES-192-CBC' and 'AES-256-CBC'. Default value: 'AES-256-CBC'.
- `phase_2_encryption` - (Optional) Phase two Encryption. Valid values: '3DES', 'AES-128-CBC', 'AES-192-CBC', 'AES-256-CBC', 'AES-128-GCM-64', 'AES-128-GCM-96' and 'AES-128-GCM-128'. Default value: 'AES-256-CBC'.
- `private_route_encryption` - (Optional) Private route encryption switch. Valid values: true, false.
- `route_table_list` - (Optional) Route tables to modify.
- `remote_gateway_latitude` - (Optional) Latitude of remote gateway. Does not support refresh.
- `remote_gateway_longitude` - (Optional) Longitude of remote gateway. Does not support refresh.
- `backup_remote_gateway_latitude` - (Optional) Latitude of backup remote gateway. Does not support refresh.
- `backup_remote_gateway_longitude` - (Optional) Longitude of backup remote gateway. Does not support refresh.
- `ssl_server_pool` - (Optional) Specify ssl_server_pool for tunnel_type "tcp". Default value: "192.168.44.0/24".
- `enable_dead_peer_detection` - (Optional) Switch to Enable/Disable Deed Peer Detection for an existing site2cloud connection. Default value: true.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `local_subnet_cidr` - Local subnet CIDR.

**NOTE:** `custom_algorithms` - Only supported for 'udp' tunnel type. If set to true, the six algorithm arguments cannot all be default value. If set to false, default values will be used for all six algorithm arguments.

**NOTE:** `ssl_server_pool` - Only supported for 'tcp' tunnel type. If not set, default value will be used. If set, needs to be set to a different value than default value.

**NOTE:** `enable_dead_peer_detection` - If you are using/upgraded to Aviatrix Terraform Provider R1.9+, and an site2cloud resource was originally created with a provider version <R1.9, you must do 'terraform refresh' to update and apply the attribute's default value (true) into the state file.

## Import

Instance site2cloud can be imported using the connection_name and vpc_id, e.g.

```
$ terraform import aviatrix_site2cloud.test connection_name~vpc_id
```

# aviatrix_spoke_gateway

The aviatrix_spoke_gateway resource allows to create and manage Aviatrix Spoke Gateways.

## Example Usage

```
# Create an Aviatrix AWS Spoke Gateway
resource "aviatrix_spoke_gateway" "test_spoke_gateway_aws" {
  cloud_type   = 1
  account_name = "my-aws"
  gw_name      = "spoke-gw-aws"
  vpc_id       = "vpc-abcd123"
  vpc_reg      = "us-west-1"
  gw_size      = "t2.micro"
  subnet       = "10.11.0.0/24"
  enable_snat  = false
  tag_list     = [
    "k1:v1",
    "k2:v2",
  ]
}

# Create an Aviatrix GCP Spoke Gateway
resource "aviatrix_spoke_gateway" "test_spoke_gateway_gcp" {
  cloud_type   = 4
  account_name = "my-gcp"
  gw_name      = "spoke-gw-gcp"
  vpc_id       = "gcp-spoke-vpc"
  vpc_reg      = "us-west1-b"
  gw_size      = "n1-standard-1"
  subnet       = "10.12.0.0/24"
  enable_snat  = false
}

# Create an Aviatrix ARM Spoke Gateway
resource "aviatrix_spoke_gateway" "test_spoke_gateway_arm" {
  cloud_type   = 8
  account_name = "my-arm"
  gw_name      = "spoke-gw-01"
  vpc_id       = "spoke:test-spoke-gw-123"
  vpc_reg      = "West US"
  gw_size      = "Standard_B1s"
  subnet       = "10.13.0.0/24"
  enable_snat  = false
}

# Create an Aviatrix Oracle Spoke Gateway
resource "aviatrix_spoke_gateway" "test_spoke_gateway_oracle" {
  cloud_type   = 16
  account_name = "devops-oracle"
  gw_name      = "avtxgw-oracle"
  vpc_id       = "vpc-oracle-test"
  vpc_reg      = "us-ashburn-1"
  gw_size      = "VM.Standard2.2"
  subnet       = "10.7.0.0/16"
}
```

## Argument Reference

The following arguments are supported:

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1), GCP(4), ARM(8), and OCI(16) are supported.
- `account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `gw_name` - (Required) Name of the gateway which is going to be created.
- `vpc_id` - (Required) VPC-ID/VNet-Name of cloud provider. Required if for aws. Example: AWS: "vpc-abcd1234", GCP: "vpc-gcp-test", ARM: "vnet1:hello", OCI: "vpc-oracle-test1".
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", GCP: "us-west2-a", ARM: "East US 2", Oracle: "us-ashburn-1".
- `gw_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", ARM: "Standard_B1s", Oracle: "VM.Standard2.2", GCP: "n1-standard-1".
- `subnet` - (Required) A VPC Network address range selected from one of the available network ranges. Example: "172.31.0.0/20".
- `insane_mode_az` - (Required) AZ of subnet being created for Insane Mode Spoke Gateway. Required for AWS if insane_mode is enabled. Example: AWS: "us-west-1a".
- `allocate_new_eip` - (Optional) When value is false, reuse an idle address in Elastic IP pool for this gateway. Otherwise, allocate a new Elastic IP and use it for this gateway. Available in controller 4.7 or later release. Supported values: true, false. Default: true. Option not available for GCP, ARM and Oracle gateways, they will automatically allocate new eip's.
- `eip` - (Optional) Required when allocate_new_eip is false. It uses specified EIP for this gateway. Available in controller 4.7 or later release.
- `ha_subnet` - (Optional) HA Subnet. Required for enabling HA for AWS/ARM gateway. Setting to empty/unset will disable HA. Setting to a valid subnet CIDR will create an HA gateway on the subnet. Example: "10.12.0.0/24"
- `ha_zone` - (Optional) HA Zone. Required for enabling HA for GCP gateway. Setting to empty/unset will disable HA. Setting to a valid zone will create an HA gateway in the zone. Example: "us-west1-c".
- `ha_insane_mode_az` (Optional) AZ of subnet being created for Insane Mode Spoke HA Gateway. Required for AWS if insane_mode is enabled and ha_subnet is set. Example: AWS: "us-west-1a".
- `ha_gw_size` - (Optional) HA Gateway Size. Mandatory if HA is enabled (ha_subnet is set). Example: "t2.micro".
- `ha_eip` - (Optional) Public IP address that you want to assign to the HA peering instance. If no value is given, a new eip will automatically allocated. Only available for AWS.
- `enable_snat` - (Optional) Specify whether enabling Source NAT feature on the gateway or not. Please disable AWS NAT instance before enabling this feature. Supported values: true, false.
- `single_az_ha` (Optional) Set to true if this feature is desired. Supported values: true, false.
- `transit_gw` - (Optional) Specify the transit Gateway.
- `tag_list` - (Optional) Instance tag of cloud provider. Only AWS, cloud_type is "1", is supported. Example: ["key1:value1", "key2:value2"].
- `insane_mode` - (Optional) Enable Insane Mode for Spoke Gateway. Insane Mode gateway size has to be at least c5 (AWS) or Standard_D3_v2 (ARM). If enabled, will look for spare /26 segment to create a new subnet. Only supported for AWS and ARM. Supported values: true, false.

- `enable_active_mesh` - (Optional) Switch to Enable/Disable Active Mesh Mode for Spoke Gateway. Valid values: true, false. Default value: false.
- `enable_vpc_dns_server` - (Optional) Enable VPC DNS Server for Gateway. Currently only supports AWS. Valid values: true, false. Default value: false.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `eip` - Public IP address assigned to the gateway.
- `ha_eip` - Public IP address assigned to the HA gateway.
- `cloud_instance_id` - Cloud Instance ID.

## Import

Instance spoke_gateway can be imported using the gw_name, e.g.

```
$ terraform import aviatrix_spoke_gateway.test gw_name
```

# aviatrix_spoke_vpc

The aviatrix_spoke_vpc resource allows to create and manage Aviatrix Spoke Gateways.

**NOTE:** The spoke_Vpc resource is deprecated. It is kept for backward compatibility and will be removed in the future. Please use spoke_gateway instead. Need to remove it from state file and import as aviatrix_spoke_gateway if it is already in state.

## Example Usage

```
# Set Aviatrix aws spoke_vpc
resource "aviatrix_spoke_vpc" "test_spoke_vpc_aws" {
  cloud_type   = 1
  account_name = "my-aws"
  gw_name      = "spoke-gw-aws"
  vpc_id       = "vpc-abcd123~~spoke-vpc-01"
  vpc_reg      = "us-west-1"
  vpc_size     = "t2.micro"
  subnet       = "10.11.0.0/24~~us-west-1b~~spoke-vpc-01-pubsub"
  enable_nat   = "no"
  dns_server   = "8.8.8.8"
  tag_list     = [
    "k1:v1",
    "k2:v2",
  ]
}

# Set Aviatrix gcp spoke_vpc
resource "aviatrix_spoke_vpc" "test_spoke_vpc_gcp" {
  cloud_type   = 4
  account_name = "my-gcp"
  gw_name      = "spoke-gw-gcp"
  vpc_id       = "gcp-spoke-vpc"
  vpc_reg      = "us-west1-b"
  vpc_size     = "t2.micro"
  subnet       = "10.12.0.0/24"
  enable_nat   = "no"
}

# Set Aviatrix arm spoke_vpc
resource "aviatrix_spoke_vpc" "test_spoke_vpc_arm" {
  cloud_type   = 8
  account_name = "my-arm"
  gw_name      = "spoke-gw-01"
  vpc_id       = "spoke:test-spoke-gw-123"
  vpc_reg      = "West US"
  vpc_size     = "t2.micro"
  subnet       = "10.13.0.0/24"
  enable_nat   = "no"
}
```

# Argument Reference

The following arguments are supported:

- `cloud_type` - (Required) Type of cloud service provider. AWS=1, GCP=4, ARM=8.
- `account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `gw_name` - (Required) Name of the gateway which is going to be created.
- `vpc_id` - (Required) VPC-ID/VNet-Name of cloud provider. Required if cloud_type is "1" or "4". Example: AWS: "vpc-abcd1234", etc...
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", GCP: "us-west1-b", ARM: "East US 2", etc...
- `vpc_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", GCP: "f1.micro", ARM: "StandardD2", etc...
- `subnet` - (Required) Public Subnet Info. Example: AWS: "CIDRZONESubnetName", etc...
- `ha_subnet` - (Optional) HA Subnet. Required for enabling HA for AWS/ARM gateways. Setting to empty/unset will disable HA. Setting to a valid subnet (Example: 10.12.0.0/24) will create an HA gateway on the subnet.
- `ha_zone` - (Optional) HA Zone. Required for enabling HA for GCP gateway. Setting to empty/unset will disable HA. Setting to a valid zone will create an HA gateway in the zone. Example: "us-west1-c".
- `ha_gw_size` - (Optional) HA Gateway Size. Mandatory if HA is enabled (ha_subnet is set). Example: "t2.micro".
- `enable_snat` - (Optional) Enable Source NAT for this container. Supported values: true, false. Default value: false.
- `single_az_ha` - (Optional) Set to "enabled" if this feature is desired.
- `transit_gw` - (Optional) Specify the transit Gateway.
- `tag_list` - (Optional) Instance tag of cloud provider. Example: key1:value1,key002:value002, etc... Only AWS (cloud_type is "1") is supported

The following arguments are deprecated:

- `dns_server` - Specify the DNS IP, only required while using a custom private DNS for the VPC.

**NOTE:** `vnet_and_resource_group_names` - If you are using/upgraded to Aviatrix Terraform Provider R1.10+, and an ARM spoke_vpc resource was originally created with a provider version < R1.10, you must replace "vnet_and_resource_group_names" with "vpc_id" in your configuration file, and do 'terraform refresh' to set its value to "vpc_id" and apply it into the state file.

# Import

Instance spoke_vpc can be imported using the gw_name, e.g.

```
$ terraform import aviatrix_spoke_vpc.test gw_name
```

# aviatrix_transit_gateway

The aviatrix_transit_gateway resource creates and manages the Aviatrix Transit Network Gateways.

## Example Usage

```
# Create an Aviatrix AWS Transit Network Gateway
resource "aviatrix_transit_gateway" "test_transit_gateway_aws" {
  cloud_type               = 1
  account_name             = "devops_aws"
  gw_name                  = "transit"
  vpc_id                   = "vpc-abcd1234"
  vpc_reg                  = "us-east-1"
  gw_size                  = "t2.micro"
  subnet                   = "10.1.0.0/24"
  ha_subnet                = "10.1.0.0/24"
  ha_gw_size               = "t2.micro"
  tag_list                 = [
    "name:value",
    "name1:value1",
    "name2:value2",
  ]
  enable_hybrid_connection = true
  connected_transit        = true
}

# Create an Aviatrix GCP Transit Network Gateway
resource "aviatrix_transit_gateway" "test_transit_gateway_gcp" {
  cloud_type   = 4
  account_name = "devops-gcp"
  gw_name      = "avtxgw-gcp"
  vpc_id       = "vpc-gcp-test"
  vpc_reg      = "us-west2-a"
  gw_size      = "n1-standard-1"
  subnet       = "10.8.0.0/16"
  ha_zone      = "us-west2-b"
  ha_gw_size   = "n1-standard-1"
}

# Create an Aviatrix ARM Transit Network Gateway
resource "aviatrix_transit_gateway" "test_transit_gateway_azure" {
  cloud_type        = 8
  account_name      = "devops_azure"
  gw_name           = "transit"
  vpc_id            = "vnet1:hello"
  vpc_reg           = "West US"
  gw_size           = "Standard_B1s"
  subnet            = "10.30.0.0/24"
  ha_subnet         = "10.30.0.0/24"
  ha_gw_size        = "Standard_B1s"
  connected_transit = true
}

# Create an Aviatrix Oracle Transit Network Gateway
resource "aviatrix_transit_gateway" "test_transit_gateway_oracle" {
  cloud_type   = 16
  account_name = "devops-oracle"
  gw_name      = "avtxgw-oracle"
  vpc_id       = "vpc-oracle-test"
  vpc_reg      = "us-ashburn-1"
  gw_size      = "VM.Standard2.2"
  subnet       = "10.7.0.0/16"
}
```

# Argument Reference

The following arguments are supported:

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1), GCP(4), ARM(8), and OCI(16) are supported.
- `account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `gw_name` - (Required) Name of the gateway which is going to be created.
- `vpc_id` - (Required) VPC-ID/VNet-Name of cloud provider. Required if for aws. Example: AWS: "vpc-abcd1234", GCP: "vpc-gcp-test".
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", ARM: "East US 2", Oracle: "us-ashburn-1", GCP: "us-west2-a".
- `gw_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", ARM: "Standard_B1s", Oracle: "VM.Standard2.2", GCP: "n1-standard-1".
- `subnet` - (Required) A VPC Network address range selected from one of the available network ranges. Example: "172.31.0.0/20".
- `allocate_new_eip` - (Optional) When value is false, reuse an idle address in Elastic IP pool for this gateway. Otherwise, allocate a new Elastic IP and use it for this gateway. Available in controller 4.7 or later release. Supported values: true, false. Default: true. Option not available for GCP, ARM and Oracle gateways, they will automatically allocate new eip's.
- `eip` - (Optional) Required when allocate_new_eip is false. It uses specified EIP for this gateway. Available in controller 4.7 or later release.
- `ha_subnet` - (Optional) HA Subnet CIDR. Required for enabling HA for AWS/ARM gateway. Setting to empty/unset will disable HA. Setting to a valid subnet CIDR will create an HA gateway on the subnet. Example: "10.12.0.0/24".
- `ha_zone` - (Optional) HA Zone. Required for enabling HA for GCP gateway. Setting to empty/unset will disable HA. Setting to a valid zone will create an HA gateway in the zone. Example: "us-west1-c".
- `ha_gw_size` - (Optional) HA Gateway Size. Mandatory if HA is enabled (ha_subnet is set). Example: "t2.micro".
- `ha_eip` - (Optional) Public IP address that you want to assign to the HA peering instance. If no value is given, a new eip will automatically allocated. Only available for AWS.
- `enable_snat` - (Optional) Enable Source NAT for this container. Supported values: true, false.
- `single_az_ha` (Optional) Set to true if this feature is desired. Supported values: true, false.
- `tag_list` - (Optional) Instance tag of cloud provider. Only supported for AWS. Example: ["key1:value1","key2:value2"].
- `enable_hybrid_connection` - (Optional) Sign of readiness for TGW connection. Only supported for AWS. Example: false.
- `enable_firenet` - (Optional) Sign of readiness for FireNet connection. Valid values: true, false. Default value: false.
- `connected_transit` - (Optional) Specify Connected Transit status. If enabled, it allows spokes to run traffics to other spokes via transit gateway. Supported values: true, false. Default value: false.

- `insane_mode` - (Optional) Specify Insane Mode high performance gateway. Insane Mode gateway size must be at least c5 size (AWS) or Standard_D3_v2 (ARM). If enabled, will look for spare /26 segment to create a new subnet. Only available for AWS and ARM. Supported values: true, false.
- `insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Transit Gateway. Required for AWS if insane_mode is enabled. Example: AWS: "us-west-1a".
- `ha_insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Transit HA Gateway. Required for AWS if insane_mode is enabled and ha_subnet is set. Example: AWS: "us-west-1a".
- `enable_active_mesh` - (Optional) Switch to Enable/Disable Active Mesh Mode for Transit Gateway. Valid values: true, false. Default value: false.
- `enable_vpc_dns_server` - (Optional) Enable VPC DNS Server for Gateway. Currently only supports AWS. Valid values: true, false. Default value: false.
- `enable_advertise_transit_cidr` - (Optional) Switch to Enable/Disable advertise transit VPC network CIDR for a vgw connection. Available as of R2.6.
- `bgp_manual_spoke_advertise_cidrs` - (Optional) Intended CIDR list to advertise to VGW. Example: "10.2.0.0/16,10.4.0.0/16". Available as of R2.6.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `eip` - Public IP address assigned to the gateway.
- `ha_eip` - Public IP address assigned to the HA gateway.

The following arguments are deprecated:

- `enable_firenet_interfaces` - (Optional) Sign of readiness for FireNet connection. Valid values: true, false. Default value: false.

**NOTE:**

- `enable_firenet` - If you are using/upgraded to Aviatrix Terraform Provider R2.5+/UserConnect-5.0+ , and an AWS transit_gateway resource with "enable_firenet_interfaces" enabled was created with a provider version < R2.5/ UserConnect-5.0, you must replace "enable_firenet_interfaces" with "enable_firenet" in your configuration file, and do ‘terraform refresh’ to set its value to "enable_firenet" and apply it into the state file.
- `enable_advertise_transit_cidr` and `bgp_manual_spoke_advertise_cidrs` functionality has been migrated over to **aviatrix_transit_gateway** as of Aviatrix Terraform Provider R2.6. If you are using/upgraded to Aviatrix Terraform Provider R2.6+, and a vgw_conn resource was originally created with a provider version <R2.6, you must cut and paste these two arguments (and values) into the corresponding transit gateway resource referenced in the **vgw_conn**. A 'terraform refresh' will then successfully complete the migration and rectify the state file.

## Import

Instance transit_gateway can be imported using the gw_name, e.g.

```
$ terraform import aviatrix_transit_gateway.test gw_name
```

# aviatrix_transit_gateway_peering

The aviatrix_transit_gateway_peering resource allows the creation and management of Aviatrix Transit Gateway Peerings.

## Example Usage

```
# Create an Aviatrix Transit Gateway Peering
resource "aviatrix_transit_gateway_peering" "test_transit_gateway_peering" {
  transit_gateway_name1 = "transitGw1"
  transit_gateway_name2 = "transitGw2"
}
```

## Argument Reference

The following arguments are supported:

- `transit_gateway_name1` - (Required) The first transit gateway name to make a peer pair.
- `transit_gateway_name2` - (Required) The second transit gateway name to make a peer pair.

## Import

Instance transit_vpc can be imported using the transit_gateway_name1 and transit_gateway_name2, e.g.

```
$ terraform import aviatrix_transit_gateway_peering.test transit_gateway_name1~transit_gateway_name2
```

# aviatrix_transit_vpc

The aviatrix_transit_vpc resource creates and manages the Aviatrix Transit Network Gateways.

**NOTE:** The aviatrix_transit_vpc resource is deprecated. It is kept for backward compatibility and will be removed in the future. Please use transit_gateway instead. Need to remove it from state file and import as aviatrix_transit_gateway if it is already in state.

## Example Usage

```
# Manage Aviatrix Transit Network Gateways in aws
resource "aviatrix_transit_vpc" "test_transit_gw_aws" {
  cloud_type               = 1
  account_name             = "devops_aws"
  gw_name                  = "transit"
  vpc_id                   = "vpc-abcd1234"
  vpc_reg                  = "us-east-1"
  vpc_size                 = "t2.micro"
  subnet                   = "10.1.0.0/24"
  ha_subnet                = "10.1.0.0/24"
  ha_gw_size               = "t2.micro"
  tag_list                 = [
    "name:value",
    "name1:value1",
    "name2:value2"
  ]
  enable_hybrid_connection = true
  connected_transit        = "yes"
}

# Manage Aviatrix Transit Network Gateways in azure
resource "aviatrix_transit_vpc" "test_transit_gw_azure" {
  cloud_type        = 8
  account_name      = "devops_azure"
  gw_name           = "transit"
  vpc_id            = "vnet1:hello"
  vpc_reg           = "West US"
  vpc_size          = "Standard_B1s"
  subnet            = "10.30.0.0/24"
  ha_subnet         = "10.30.0.0/24"
  ha_gw_size        = "Standard_B1s"
  connected_transit = "yes"
}
```

## Argument Reference

The following arguments are supported:

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Use 1 for AWS.
- `account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `gw_name` - (Required) Name of the gateway which is going to be created.
- `vpc_id` - (Required) VPC-ID/VNet-Name of cloud provider. Required if for aws. Example: AWS: "vpc-abcd1234", GCP: "mygooglecloudvpcname", etc...
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", ARM: "East US 2", etc...
- `vpc_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", etc...
- `subnet` - (Required) Public Subnet CIDR. Example: AWS: "10.0.0.0/24". Copy/paste from AWS Console to get the right subnet CIDR.
- `ha_subnet` - (Optional) HA Subnet CIDR. Example: "10.12.0.0/24".Setting to empty/unset will disable HA. Setting to a valid subnet CIDR will create an HA gateway on the subnet.
- `ha_gw_size` - (Optional) HA Gateway Size. Mandatory if HA is enabled (ha_subnet is set). Example: "t2.micro".
- `enable_snat` - (Optional) Enable Source NAT for this container. Supported values: true, false. Default value: false.
- `tag_list` - (Optional) Instance tag of cloud provider. Only supported for aws. Example: ["key1:value1","key002:value002"]
- `enable_hybrid_connection` - (Optional) Sign of readiness for TGW connection. Only supported for aws. Example: false.
- `enable_firenet_interfaces` - (Optional) Sign of readiness for FireNet connection. Valid values: true and false. Default: false.
- `connected_transit` - (Optional) Specify Connected Transit status. Supported values: true, false.
- `insane_mode` - (Optional) Specify Insane Mode high performance gateway. Insane Mode gateway size must be at least c5 size. If enabled, will look for spare /26 segment to create a new subnet. Only available for AWS. Supported values: true, false.
- `insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Transit Gateway. Required if insane_mode is enabled.
- `ha_insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Transit HA Gateway. Required if insane_mode is enabled and ha_subnet is set.

The following arguments are deprecated:

- `dns_server` - Specify the DNS IP, only required while using a custom private DNS for the VPC.
- `vnet_name_resource_group` - (Optional) VPC-ID/VNet-Name of cloud provider. Required if for azure. ARM: "VNet_Name:Resource_Group_Name". It is replaced by "vpc_id".

**NOTE:** `enable_firenet_interfaces` - If you are using/upgraded to Aviatrix Terraform Provider R1.8+, and a transit_vpc resource was originally created with a provider version < R1.8, you must do 'terraform refresh' to update and apply the attribute's default value (false) into the state file.

**NOTE:** `vnet_name_resource_group` - If you are using/upgraded to Aviatrix Terraform Provider R1.10+, and an ARM

transit_vpc resource was originally created with a provider version < R1.10, you must replace "vnet_name_resource_group" with "vpc_id" in your configuration file, and do 'terraform refresh' to set its value to "vpc_id" and apply it into the state file.

## Import

Instance transit_vpc can be imported using the gw_name, e.g.

```
$ terraform import aviatrix_transit_vpc.test gw_name
```

# aviatrix_trans_peer

The aviatrix_trans_peer resource allows the creation and management of Aviatrix Transitive Peerings.

## Example Usage

```
# Create an Aviatrix AWS Transitive Peering
resource "aviatrix_trans_peer" "test_trans_peer" {
  source         = "avtxuseastgw1"
  nexthop        = "avtxuseastgw2"
  reachable_cidr = "10.152.0.0/16"
}
```

## Argument Reference

The following arguments are supported:

- `source` - (Required) Name of Source gateway.
- `nexthop` - (Required) Name of nexthop gateway.
- `reachable_cidr` - (Required) Destination CIDR.

## Import

Instance trans_peer can be imported using the source, nexthop and reachable_cidr, e.g.

```
$ terraform import aviatrix_trans_peer.test source~nexthop~reachable_cidr
```

# aviatrix_tunnel

The aviatrix_tunnel resource allows the creation and management of Aviatrix tunnels.

## Example Usage

```
# Create an Aviatrix AWS Tunnel
resource "aviatrix_tunnel" "test_tunnel" {
  gw_name1 = "avtxgw1"
  gw_name2 = "avtxgw2"
}
```

## Argument Reference

The following arguments are supported:

- `gw_name1` - (Required) The first VPC Container name to make a peer pair.
- `gw_name2` - (Required) The second VPC Container name to make a peer pair.
- `enable_ha` - (Optional) Whether Peering HA is enabled. Valid values: true, false.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `peering_state` - (Computed) Status of the tunnel.
- `peering_hastatus` - (Computed) Status of the HA tunnel.
- `peering_link` - (Computed) Name of the peering link.

## Import

Instance tunnel can be imported using the gw_name1 and gw_name2, e.g.

```
$ terraform import aviatrix_tunnel.test gw_name1~gw_name2
```

# aviatrix_vgw_conn

The aviatrix_vgw_conn resource managers the Aviatrix Transit Gateway to VGW Connection.

## Example Usage

```
# Create an Aviatrix Vgw Connection
resource "aviatrix_vgw_conn" "test_vgw_conn" {
  conn_name        = "my-connection-vgw-to-tgw"
  gw_name          = "my-transit-gw"
  vpc_id           = "vpc-abcd1234"
  bgp_vgw_id       = "vgw-abcd1234"
  bgp_vgw_account  = "dev-account-1"
  bgp_vgw_region   = "us-east-1"
  bgp_local_as_num = "65001"
}
```

## Argument Reference

The following arguments are supported:

- `conn_name` - (Required) The name of for Transit GW to VGW connection connection which is going to be created. Example: "my-connection-vgw-to-tgw".
- `gw_name` - (Required) Name of the Transit Gateway. Example: "my-transit-gw".
- `vpc_id` - (Required) VPC-ID where the Transit Gateway is located. Example: AWS: "vpc-abcd1234".
- `bgp_vgw_id` - (Required) Id of AWS's VGW that is used for this connection. Example: "vgw-abcd1234".
- `bgp_vgw_account` - (Required) Account of AWS's VGW that is used for this connection. Example: "dev-account-1".
- `bgp_vgw_region` - (Required) Region of AWS's VGW that is used for this connection. Example: "us-east-1".
- `bgp_local_as_num` - (Required) BGP Local ASN (Autonomous System Number). Integer between 1-65535. Example: "65001".

The following arguments are deprecated:

- `enable_advertise_transit_cidr` - (Optional) Switch to Enable/Disable advertise transit VPC network CIDR for a vgw connection.
- `bgp_manual_spoke_advertise_cidrs` - (Optional) Intended CIDR list to advertise to VGW. Example: "10.2.0.0/16,10.4.0.0/16".

**NOTE:** `enable_advertise_transit_cidr` - If you are using/upgraded to Aviatrix Terraform Provider R1.9+, and a vgw_conn resource was originally created with a provider version <R1.9, you must do 'terraform refresh' to update and apply the attribute's default value (false) into the state file.

**NOTE:** `enable_advertise_transit_cidr` and `bgp_manual_spoke_advertise_cidrs` functionality has been migrated over to **aviatrix_transit_gateway** as of Aviatrix Terraform Provider R2.6. If you are using/upgraded to Aviatrix Terraform Provider R2.6+, and a vgw_conn resource was originally created with a provider version <R2.6, you must cut and paste these two arguments (and values) into the corresponding transit gateway resource referenced in this **vgw_conn**. A 'terraform refresh' will then successfully complete the migration and rectify the state file.

## Import

Instance vgw_conn can be imported using the conn_name and vpc_id, e.g.

```
$ terraform import aviatrix_vgw_conn.test conn_name~vpc_id
```

# aviatrix_vpc

The aviatrix_vpc resource allows the creation and management of VPCs.

## Example Usage

```
# Create an AWS VPC
resource "aviatrix_vpc" "test_vpc" {
  cloud_type           = 1
  account_name         = "devops"
  region               = "us-west-1"
  name                 = "vpcTest"
  cidr                 = "10.0.0.0/16"
  aviatrix_transit_vpc = false
  aviatrix_firenet_vpc = false
}
```

## Argument Reference

The following arguments are supported:

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1) is supported.
- `account_name` - (Required) This parameter represents the name of a Cloud-Account in Aviatrix controller.
- `name` - (Required) Name of the vpc to be created.
- `region` - (Required) Region of cloud provider. Example: AWS: "us-east-1", ARM: "East US 2".
- `cidr` - (Required) VPC cidr.
- `aviatrix_transit_vpc` - (Optional) Specify whether it is an Aviatrix Transit VPC. Valid values: true, false. Default: false.
- `aviatrix_firenet_vpc` - (Optional) Specify whether it is an Aviatrix Firenet VPC. Valid values: true, false. Default: false.

## Attribute Reference

In addition to all arguments above, the following attributes are exported:

- `vpc_id` - ID of the vpc to be created.
- `subnets` - List of subnet of the VPC to be created.
  - `cidr` - CIDR block.
  - `name` - Name of this subnet.
  - `subnet_id` - ID of this subnet.

- `public_subnets` - List of public subnet of the VPC to be created.
  - `cidr` - CIDR block.
  - `name` - Name of this subnet.
  - `subnet_id` - ID of this subnet.
- `private_subnets` - List of private subnet of the VPC to be created.
  - `cidr` - CIDR block.
  - `name` - Name of this subnet.
  - `subnet_id` - ID of this subnet.

**NOTE:** `aviatrix_firenet_vpc` - If you are using/ upgraded to Aviatrix Terraform Provider R1.8+, and an vpc resource was originally created with a provider version < R1.8, you must do 'terraform refresh' to update and apply the attribute's default value (false) into the state file.

**NOTE:** `subnets` - If created as a FireNet VPC, four public subnets will be created in the following order: subnet for firewall-mgmt in the first zone, subnet for ingress-egress in the first zone, subnet for firewall-mgmt in the second zone, and subnet for ingress-egress in the second zone.

## Import

Instance vpc can be imported using the name, e.g.

```
$ terraform import aviatrix_vpc.test name
```

# aviatrix_vpn_profile

The aviatrix_vpn_profile resource allows the creation and management of Aviatrix VPN VPN User Profiles.

## Example Usage

```
# Create an Aviatrix AWS VPN User Profile
resource "aviatrix_vpn_profile" "test_vpn_profile" {
  name      = "my_profile"
  base_rule = "allow_all"
  users     = [
    "user1",
    "user2"
  ]

  policy {
    action = "deny"
    proto  = "tcp"
    port   = "443"
    target = "10.0.0.0/32"
  }

  policy {
    action = "deny"
    proto  = "tcp"
    port   = "443"
    target = "10.0.0.1/32"
  }
}
```

## Argument Reference

The following arguments are supported:

- `name` - (Required) Enter any name for the VPN profile.
- `base_rule` - (Optional) Base policy rule of the profile to be added. Enter "allow_all" or "deny_all", based on whether you want a white list or black list.
- `users` - (Optional) List of VPN users to attach to this profile.
- `policy` - (Optional) New security policy for the profile. Each policy has the following attributes:
  - `action` - (Required) Should be the opposite of the base rule for correct behaviour. Valid values for action: "allow", "deny".
  - `proto` - (Required) Protocol to allow or deny. Valid values for protocol: "all", "tcp", "udp", "icmp", "sctp", "rdp", "dccp".
  - `port` - (Required) Port to be allowed or denied. Valid values for port: a single port or a range of port numbers e.g.: "25", "25:1024". For "all" and "icmp", port should only be "0:65535".

- `target` - (Required) CIDR to be allowed or denied. Valid values for target: IPv4 CIDRs. Example: "10.30.0.0/16".

## Import

Instance vpn_profile can be imported using the name, e.g.

```
$ terraform import aviatrix_vpn_profile.test name
```

# aviatrix_vpn_user_accelerator

The aviatrix_vpn_user_accelerator resource manages the Aviatrix VPN User Accelerators.

## Example Usage

```
# Create an Aviatrix Vpn User Accelerator
resource "aviatrix_vpn_user_accelerator" "test_vpc_accelerator" {
  elb_name = "Aviatrix-vpc-abcd2134"
}
```

## Argument Reference

The following arguments are supported:

- `elb_name` - (Required) Name of ELB to be added to VPN User Accelerator. Example: "Aviatrix-vpc-abcd2134".

## Import

```
$ terraform import aviatrix_vpn_user_acclerator.test Aviatrix-vpc-abcd1234
```

# aviatrix_vpn_user

The aviatrix_vpn_user resource creates and manages VPN Users.

## Example Usage

```
# Create an Aviatrix Vpn User
resource "aviatrix_vpn_user" "test_vpn_user" {
  vpc_id     = "vpc-abcd1234"
  gw_name    = "gw1"
  user_name  = "username1"
  user_email = "user@aviatrix.com"
}
```

## Argument Reference

The following arguments are supported:

- `vpc_id` - (Required) VPC Id of Aviatrix VPN gateway. Example: "vpc-abcd1234".
- `gw_name` - (Required) If ELB is enabled, this will be the name of the ELB, else it will be the name of the Aviatrix VPN gateway. Example: "gw1".
- `user_name` - (Required) VPN user name. Example: "user".
- `user_email` - (Optional) VPN User's email. Example: "abc@xyz.com (mailto:abc@xyz.com)".
- `saml_endpoint` - (Optional) This is the name of the SAML endpoint to which the user is to be associated. This is required if adding user to a SAML gateway/LB.

## Import

Instance vpn_user can be imported using the user_name, e.g.

```
$ terraform import aviatrix_vpn_user.test user_name
```