» 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_account

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

NOTE: With the release of Controller 5.4 (compatible with Aviatrix provider R2.13), Role-Based Access Control (RBAC) is now integrated into the Accounts workflow. Any **aviatrix_account** created in 5.3 by default will have admin privileges (attached to the 'admin' RBAC permission group). In 5.4, any new accounts created will not be attached to any RBAC group unless otherwise specified through the **aviatrix_rbac_group_access_account_attachment** resource.

» Example Usage

```
# Create an Aviatrix AWS Account with IAM roles
resource "aviatrix_account" "temp_acc_aws" {
  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" "temp_acc_aws" {
  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" "temp_acc_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 Account
resource "aviatrix_account" "temp_acc_azure" {
  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" "temp_acc_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"
}

# Create an Aviatrix AWS Gov Account
resource "aviatrix_account" "temp_acc_awsgov" {
  account_name          = "username"
  cloud_type            = 256
  awsgov_account_number = "123456789012"
  awsgov_access_key     = "ABCDEFGHIJKL"
  awsgov_secret_key     = "ABCDEFGHIJKLabcdefghijkl"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `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, AZURE, OCI, and AWS Gov are supported currently. Enter 1 for AWS, 4 for GCP, 8 for AZURE, 16 for OCI, 256 for AWS Gov.

» AWS

- `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 User-Connect. 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 User-Connect. Required when `aws_iam` is "true" and when creating an account for AWS.

» Azure

- `arm_subscription_id` - (Optional) Azure ARM Subscription ID. Required when creating an account for Azure.
- `arm_directory_id` - (Optional) Azure ARM Directory ID. Required when creating an account for Azure.
- `arm_application_id` - (Optional) Azure ARM Application ID. Required when creating an account for Azure.
- `arm_application_key` - (Optional) Azure ARM Application key. Required when creating an account for Azure.

» Google Cloud

- `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.

» Oracle Cloud

- `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.

» AWS GovCloud

- `awsgov_account_number` - (Optional) AWS Gov Account number to associate with Aviatrix account. Required when creating an account for AWS Gov.
- `awsgov_access_key` - (Optional) AWS Access Key. Required when creating an account for AWS Gov.
- `awsgov_secret_key` - (Optional) AWS Secret Key. Required when creating an account for AWS Gov.

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 and https://docs.aviatrix.com/HowTos/HowTo_IAM_role.html

» Import

account can be imported using the `account_name` (when doing import, need 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.

NOTE: With the release of Controller 5.4 (compatible with Aviatrix provider R2.13), Role-Based Access Control (RBAC) is now integrated into the Accounts workflow. Any **aviatrix_account_user** created in 5.3 by default will have admin privileges (attached to the 'admin' RBAC permission group). In 5.4, any new account users created will no longer have the option to specify an `account_name`, but rather have the option to attach the user to specific RBAC groups through the **aviatrix_rbac_group_user_attachment** resource for more granular security control. Account users created in 5.4 will have minimal access (read_only) unless otherwise specified in the RBAC group permissions that the users are attached to.

» Example Usage

```
# Create an Aviatrix User Account
resource "aviatrix_account_user" "test_accountuser" {
  username     = "username1"
  email        = "username1@testdomain.com"
```

```
  password     = "passwordforuser1-1234"
}
```

» Argument Reference

The following arguments are supported for creating user account:

» Required

- `username` - (Required) Name of account 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.

The following arguments are deprecated:

- `account_name` - (Required) Cloud account name of user to be created. Deprecated as of Aviatrix provider R2.13 (Controller 5.4) due to RBAC implementation.

NOTE: `account_name` - If you are using/upgraded to Aviatrix Terraform Provider R2.13+, and an **aviatrix_account_user** resource was originally created with a provider version <R2.13, you must remove this attribute and perform a 'terraform refresh' to rectify the state file.

» Import

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

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

» aviatrix_rbac_group

The **aviatrix_rbac_group** resource allows the creation and management of Aviatrix (Role-Based Access Control) RBAC groups.

» Example Usage

```
# Create an Aviatrix RBAC Group
resource "aviatrix_rbac_group" "test_group" {
  group_name = "write_only"
```

```
}
```

» Argument Reference

The following arguments are supported:

» Required

- `group_name` - (Required) This parameter represents the name of a RBAC group to be created.

» Import

rbac_group can be imported using the `group_name`, e.g.

```
$ terraform import aviatrix_rbac_group.test group_name
```

» aviatrix_rbac_group_access_account_attachment

The **aviatrix_rbac_group_access_account_attachment** resource allows the creation and management of access account attachments to Aviatrix (Role-Based Access Control) RBAC groups.

» Example Usage

```
# Create an Aviatrix RBAC Group Access Account Attachment
resource "aviatrix_rbac_group_access_account_attachment" "test_attachment" {
  group_name          = "write_only"
  access_account_name = "account_name"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `group_name` - (Required) This parameter represents the name of a RBAC group.

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

NOTE: If "all" is specified as the value for `access_account_name`, all existing access accounts will be attached to the specified RBAC group. If "all" is set, there is no need to specify any more access accounts attachments for that RBAC group.

» Import

rbac_group_access_account_attachment can be imported using the `group_name` and `access_account_name`, e.g.

```
$ terraform import aviatrix_rbac_group_access_account_attachment.test group_name~access_acc
```

» aviatrix_rbac_group_permission_attachment

The **aviatrix_rbac_group_permission_attachment** resource allows the creation and management of permission attachments to Aviatrix (Role-Based Access Control) RBAC groups.

» Example Usage

```
# Create an Aviatrix Rbac Group Permission Attachment
resource "aviatrix_rbac_group_permission_attachment" "test_attachment" {
  group_name      = "write_only"
  permission_name = "all_write"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `group_name` - (Required) This parameter represents the name of a RBAC group.
- `permission_name` - (Required) This parameter represents the permission to attach to the RBAC group.

Valid `permission_name` values: * ”all_dashboard_write” * ”all_accounts_write” * ”all_gateway_write” * ”all_tgw_orchestrator_write” * ”all_transit_network_write”: * ”all_firewall_network_write” * ”all_cloud_wan_write” * ”all_peering_write” * ”all_site2cloud_write” * ”all_openvpn_write” * ”all_security_write” * ”all_useful_tools_write” * ”all_troubleshoot_write” * ”all_write”

NOTE: If ”all_write” is specified as the value for `permission_name`, all permissions will be attached to the specified RBAC group; there is then no need to specify any more permission attachments for that RBAC group.

» Import

rbac_group_permission_attachment can be imported using the `group_name` and `permission_name`, e.g.

```
$ terraform import aviatrix_rbac_group_permission_attachment.test group_name~permission_name
```

» aviatrix_rbac_group_user_attachment

The **aviatrix_rbac_group_user_attachment** resource allows the creation and management of user attachments to Aviatrix (Role-Based Access Control) RBAC groups.

» Example Usage

```
# Create an Aviatrix RBAC Group User Attachment
resource "aviatrix_rbac_group_user_attachment" "test_attachment" {
  group_name = "write_only"
  user_name  = "user_name"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `group_name` - (Required) This parameter represents the name of a RBAC group.
- `user_name` - (Required) Username of the account user.

» Import

rbac_group_user_attachment can be imported using the `group_name` and `user_name`, e.g.

```
$ terraform import aviatrix_rbac_group_user_attachment.test group_name~user_name
```

» 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_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      = "avtx-gw-1"
  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 Peering HA enabled
resource "aviatrix_gateway" "test_gateway_aws" {
  cloud_type         = 1
  account_name       = "devops"
  gw_name            = "avtx-gw-1"
  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 AWS Gateway with VPN enabled
resource "aviatrix_gateway" "test_vpn_gateway_aws" {
  cloud_type   = 1
  account_name = "devops"
  gw_name      = "avtx-gw-1"
  vpc_id       = "vpc-abcdef"
  vpc_reg      = "us-west-1"
  gw_size      = "t2.micro"
  subnet       = "10.0.0.0/24"
  vpn_access   = true
```

```
  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      = "avtx-gw-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 GCP Gateway with Peering HA enabled
resource "aviatrix_gateway" "test_gateway_gcp" {
  cloud_type         = 4
  account_name       = "devops-gcp"
  gw_name            = "avtx-gw-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"
}

# Create an Aviatrix Azure Gateway
resource "aviatrix_gateway" "test_gateway_azure" {
  cloud_type   = 8
  account_name = "devops-azure"
  gw_name      = "avtx-gw-azure"
  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_oci" {
  cloud_type   = 16
  account_name = "devops-oci"
  gw_name      = "avtx-gw-oci"
  vpc_id       = "vpc-oracle-test"
  vpc_reg      = "us-ashburn-1"
  gw_size      = "VM.Standard2.2"
  subnet       = "10.7.0.0/16"
```

```
}

# Create an Aviatrix AWSGov Gateway
resource "aviatrix_gateway" "test_gateway_awsgov" {
  cloud_type   = 256
  account_name = "devops-awsgov"
  gw_name      = "avtx-gw-awsgov"
  vpc_id       = "vpc-abcdef"
  vpc_reg      = "us-gov-west-1"
  gw_size      = "t2.micro"
  subnet       = "10.0.0.0/24"
  tag_list     = [
    "k1:v1",
    "k2:v2",
  ]
}
```

» Argument Reference

The following arguments are supported:

» Required

- `cloud_type` - (Required) Cloud service provider to use to launch the gateway. Requires an integer value. Currently supports AWS(1), GCP(4), AZURE(8), OCI(16), and AWSGov(256).
- `account_name` - (Required) Account name. This account will be used to launch Aviatrix gateway.
- `gw_name` - (Required) Name of the Aviatrix gateway to be created.
- `vpc_id` - (Required) VPC ID/VNet name of cloud provider. Example: AWS: "vpc-abcd1234", GCP: "vpc-gcp-test", AZURE: "vnet1:hello", OCI: "vpc-oracle-test1".
- `vpc_reg` - (Required) VPC region the gateway will be created in. Example: AWS: "us-east-1", GCP: "us-west2-a", AZURE: "East US 2", OCI: "us-ashburn-1".
- `gw_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", GCP: "n1-standard-1", AZURE: "Standard_B1s", OCI: "VM.Standard2.2".
- `subnet` - (Required) A VPC network address range selected from one of the available network ranges. Example: "172.31.0.0/20". **NOTE: If using `insane_mode`, please see notes here.**

» HA

- `single_az_ha` (Optional) If enabled, Controller monitors the health of the gateway and restarts the gateway if it becomes unreachable. Valid values: true, false. Default value: false.
- `peering_ha_subnet` - (Optional) Public subnet CIDR to create Peering HA Gateway in. Required only if enabling Peering HA for AWS/AZURE. Example: AWS: "10.0.0.0/16".
- `peering_ha_zone` - (Optional) Zone to create Peering HA Gateway in. Required only if enabling Peering HA for GCP. Example: GCP: "us-west1-c".
- `peering_ha_insane_mode_az` - (Optional) Region + Availability Zone of subnet being created for Insane Mode-enabled Peering HA Gateway. Required for AWS only if `insane_mode` is set and `peering_ha_subnet` is set. Example: AWS: "us-west-1a".
- `peering_ha_eip` - (Optional) Public IP address to be assigned to the the HA peering instance. Only available for AWS.
- `peering_ha_gw_size` - (Optional) Size of the Peering HA Gateway to be created. Required if enabling Peering HA. **NOTE: Please see notes here in regards to any deltas found in your state with the addition of this argument in R1.8.**

» Insane Mode

- `insane_mode` - (Optional) Enable Insane Mode for Gateway. Insane Mode gateway size must be at least c5 series (AWS) or Standard_D3_v2 (AZURE). If enabled, a valid /26 CIDR segment of the VPC must be specified to create a new subnet. Only supported for AWS, AWSGov or Azure. Valid values: true, false.
- `insane_mode_az` - (Optional) Region + Availability Zone of subnet being created for Insane Mode gateway. Required for AWS and AWSGov if `insane_mode` is set. Example: AWS: "us-west-1a".

» SNAT/DNAT

- `single_ip_snat` - (Optional) Enable Source NAT in "single ip" mode for this gateway. Valid values: true, false. Default value: false. **NOTE: If using SNAT for FQDN use-case, please see notes here.**

NOTE: `enable_snat` has been renamed to `single_ip_snat` in provider version R2.10. Please see notes here for more information.

NOTE: Custom DNAT support has been deprecated and functionality has been moved to **aviatrix_gateway_dnat** in provider version R2.10. Please see notes here.

» VPN Access

- `vpn_access` - (Optional) Enable user access through VPN to this gateway. Valid values: true, false.
- `vpn_cidr` - (Optional) VPN CIDR block for the gateway. 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. **NOTE: Please see notes here in regards to any deltas found in your state with the addition of this argument in R1.14.**
- `enable_elb` - (Optional) Specify whether to enable ELB or not. Not supported for OCI gateways. Valid values: true, false.
- `elb_name` - (Optional) A name for the ELB that is created. If it is not specified, a name is generated automatically.
- `vpn_protocol` - (Optional) Transport mode for VPN connection. All `cloud_types` support TCP with ELB, and UDP without ELB. AWS(1) additionally supports UDP with ELB. Valid values: "TCP", "UDP". If not specified, "TCP" will be used.

» Split Tunnel

- `split_tunnel` - (Optional) Enable/disable Split Tunnel Mode. Valid values: true, false. Default value: true. Please see here for more information on split tunnel.
- `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.

» MFA Authentication

- `otp_mode` - (Optional) Two step authentication mode. Valid values: "2" for DUO, "3" for Okta.
- `saml_enabled` - (Optional) Enable/disable SAML. This field is available in Controller version 3.3 or later release. Valid values: true, false. Default value: false.
- `enable_vpn_nat` - (Optional) Enable/disable VPN NAT. Only supported for VPN gateway. Valid values: true, false. Default value: true.
- `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) Enable/disable LDAP. Valid values: true, false. Default value: 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.

» Designated Gateway

- `enable_designated_gateway` - (Optional) Enable Designated Gateway feature for Gateway. Only supported for AWS gateways. Valid values: true, false. Default value: false. Please view documentation here for more information on this feature.
- `additional_cidrs_designated_gateway` - (Optional) A list of CIDR ranges separated by comma to configure when "Designated Gateway" feature is enabled. Example: "10.8.0.0/16,10.9.0.0/16,10.10.0.0/16".

» Encryption

- `enable_encrypt_volume` - (Optional) Enable EBS volume encryption for the gateway. Only supported for AWS gateways. Valid values: true, false. Default value: false.
- `customer_managed_keys` - (Optional and Sensitive) Customer-managed key ID.

» Misc.

- `allocate_new_eip` - (Optional) If set to false, use an available 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+. Valid values: true, false. Default: true. Option not available for GCP, Azure and OCI gateways, they will automatically allocate new EIPs.
- `eip` - (Optional) Specified EIP to use for gateway creation. Required when `allocate_new_eip` is false. Available in Controller version 3.5+. Only supported for AWS gateways.
- `tag_list` - (Optional) Tag list of the gateway instance. Only available for AWS and AWSGov gateways. Example: ["key1:value1", "key2:value2"].
- `enable_vpc_dns_server` - (Optional) Enable VPC DNS Server for gateway. Currently only supported for AWS and AWSGov gateways. Valid values: true, false. Default value: false.

» Attribute Reference

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

- `elb_dns_name` - ELB DNS name.
- `public_ip` - Public IP address of the gateway created.
- `backup_public_ip` - Public IP address of the peering HA 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` - Cloud instance ID of the gateway.
- `private_ip` - Private IP address of the gateway created.
- `peering_ha_cloud_instance_id` - Cloud instance ID of the HA gateway.

The following arguments are deprecated:

- `dns_server` - Specify the DNS IP, only required while using a custom private DNS for the VPC.
- `enable_snat` - (Optional) Enable Source NAT for this gateway. Valid values: true, false. Default value: false. **NOTE: If using SNAT for FQDN use-case, please see notes here.**
- `dnat_policy` - (Optional) Policy rule applied for enabling Destination NAT (DNAT), which allows you to change the destination to a virtual address range. Currently only supports AWS(1) and AZURE(8).
  - `src_ip` - (Optional) A source IP address range where the policy rule applies.
  - `src_port` - (Optional) A source port that the policy rule applies.

  - `dst_ip` - (Optional) A destination IP address range where the policy rule applies.
  - `dst_port` - (Optional) A destination port where the policy rule applies.
  - `protocol` - (Optional) A destination port protocol where the policy rule applies.
  - `interface` - (Optional) An output interface where the policy rule applies.
  - `connection` - (Optional) Default value: "None".
  - `mark` - (Optional) A tag or mark of a TCP session where the policy rule applies.
  - `new_src_ip` - (Optional) The changed source IP address when all specified qualifier conditions meet. One of the rule fields must be specified for this rule to take effect.
  - `new_src_port` - (Optional) The translated destination port when all specified qualifier conditions meet. One of the rule field must be specified for this rule to take effect.
  - `exclude_rtb` - (Optional) This field specifies which VPC private route table will not be programmed with the default route entry.
- `cloudn_bkup_gateway_inst_id` - Instance ID of the backup gateway.

» Import

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

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

» Notes

» FQDN

In order for the FQDN feature to be enabled for the specified gateway, `single_ip_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 `single_ip_snat = true` in their config file.

» insane_mode

If `insane_mode` is enabled, you must specify a valid /26 CIDR segment of the VPC specified for the `subnet`. This will then create a new subnet to be used for the corresponding gateway. You **cannot** specify an existing /26 subnet.

» 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.

» 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.

» enable_snat

If you are using/upgraded to Aviatrix Terraform Provider R2.10+, and a gateway with `enable_snat` set to true was originally created with a provider version <R2.10, you must do a 'terraform refresh' to update and apply the attribute's value into the state. In addition, you must also change this attribute to `single_ip_snat` in your `.tf` file.

» dnat_policy

If you are using/upgraded to Aviatrix Terraform Provider R2.10+, and a gateway with `dnat_policy` was originally created with a provider version <R2.10, you must do a 'terraform refresh' to remove attribute's value from the state. In addition, you must transfer its corresponding values to the **aviatrix_gateway_dnat** resource in your `.tf` file and perform a 'terraform import' to rectify the state file.

» aviatrix_gateway_dnat

The **aviatrix_gateway_dnat** resource configures and manages policies for destination NAT function for Aviatrix gateways.

» Example Usage

```
# Add policy for destination NAT function for an Aviatrix AWS Spoke Gateway
```

```
resource "aviatrix_gateway_dnat" "test_dnat" {
  gw_name   = "avtx-gw-1"
  dnat_policy {
    src_cidr    = "13.0.0.0/16"
    src_port    = "22"
    dst_cidr    = "14.0.0.0/16"
    dst_port    = "222"
    protocol    = "tcp"
    interface   = "eth0"
    connection  = "None"
    mark        = "22"
    dnat_ips    = "175.32.12.12"
    dnat_port   = "12"
    exclude_rtb = ""
  }
}
```

» Argument Reference

The following arguments are supported:

- `gw_name` - (Required) Name of the Aviatrix gateway the custom DNAT will be configured for.
- `dnat_policy` - (Required) Policy rule applied for enabling Destination NAT (DNAT), which allows you to change the destination to a virtual address range. Currently only supports AWS(1) and AZURE(8).
  - `src_cidr` - (Optional) This is a qualifier condition that specifies a source IP address range where the rule applies. When left blank, this field is not used.
  - `src_port` - (Optional) This is a qualifier condition that specifies a source port that the rule applies. When left blank, this field is not used.
  - `dst_cidr` - (Optional) This is a qualifier condition that specifies a destination IP address range where the rule applies. When left blank, this field is not used.
  - `dst_port` - (Optional) This is a qualifier condition that specifies a destination port where the rule applies. When left blank, this field is not used.
  - `protocol` - (Optional) This is a qualifier condition that specifies a destination port protocol where the rule applies. When left blank, this field is not used.
  - `interface` - (Optional) This is a qualifier condition that specifies output interface where the rule applies. When left blank, this field is not used.
  - `connection` - (Optional) Default value: "None".

- `mark` - (Optional) This is a rule field that specifies a tag or mark of a TCP session when all qualifier conditions meet. When left blank, this field is not used.
- `dnat_ips` - (Optional) This is a rule field that specifies the translated destination IP address when all specified qualifier conditions meet. When left blank, this field is not used. One of the rule field must be specified for this rule to take effect.
- `dnat_port` - (Optional) This is a rule field that specifies the translated destination port when all specified qualifier conditions meet. When left blank, this field is not used. One of the rule field must be specified for this rule to take effect.
- `exclude_rtb` - (Optional) This field specifies which VPC private route table will not be programmed with the default route entry.

» Import

gateway_dnat can be imported using the `gw_name`, e.g.

```
$ terraform import aviatrix_gateway_dnat.test gw_name
```

» aviatrix_gateway_snat

The **aviatrix_gateway_snat** resource configures and manages policies for customized source NAT for Aviatrix gateways.

» Example Usage

```
# Enable NAT function of mode "customized_snat" for an Aviatrix AWS Spoke Gateway
resource "aviatrix_gateway_snat" "test_snat" {
  gw_name   = "avtx-gw-1"
  snat_mode = "customized_snat"
  snat_policy {
    src_cidr    = "13.0.0.0/16"
    src_port    = "22"
    dst_cidr    = "14.0.0.0/16"
    dst_port    = "222"
    protocol    = "tcp"
    interface   = "eth0"
    connection  = "None"
    mark        = "22"
    snat_ips    = "175.32.12.12"
    snat_port   = "12"
    exclude_rtb = ""
```

```
  }
}
```

» Argument Reference

The following arguments are supported:

- `gw_name` - (Required) Name of the Aviatrix gateway the custom SNAT will be configured for.
- `snat_mode` - (Optional) NAT mode. Valid values: "customized_snat". Default value: "customized_snat".
- `snat_policy` - (Required) Policy rule applied for enabling source NAT (mode: "customized_snat"). Currently only supports AWS(1) and Azure(8).
  - `src_cidr` - (Optional) This is a qualifier condition that specifies a source IP address range where the rule applies. When left blank, this field is not used.
  - `src_port` - (Optional) This is a qualifier condition that specifies a source port that the rule applies. When left blank, this field is not used.
  - `dst_cidr` - (Optional) This is a qualifier condition that specifies a destination IP address range where the rule applies. When left blank, this field is not used.
  - `dst_port` - (Optional) This is a qualifier condition that specifies a destination port where the rule applies. When left blank, this field is not used.
  - `protocol` - (Optional) This is a qualifier condition that specifies a destination port protocol where the rule applies. When left blank, this field is not used.
  - `interface` - (Optional) This is a qualifier condition that specifies output interface where the rule applies. When left blank, this field is not used.
  - `connection` - (Optional) Default value: "None".
  - `mark` - (Optional) This is a qualifier condition that specifies a tag or mark of a TCP session where the rule applies. When left blank, this field is not used.
  - `snat_ips` - (Optional) This is a rule field that specifies the changed source IP address when all specified qualifier conditions meet. When left blank, this field is not used. One of the rule fields must be specified for this rule to take effect.
  - `snat_port` - (Optional) This is a rule field that specifies the changed source port when all specified qualifier conditions meet. When left blank, this field is not used. One of the rule fields must be specified for this rule to take effect.
  - `exclude_rtb` - (Optional) This field specifies which VPC private

route table will not be programmed with the default route entry.

» Import

gateway_snat can be imported using the `gw_name`, e.g.

```
$ terraform import aviatrix_gateway_snat.test gw_name
```

» aviatrix_aws_tgw

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

NOTE: If you are planning to attach VPCs to the **aviatrix_aws_tgw** resource and anticipate updating it often and/or using advanced options such as customized route advertisement, we highly recommend managing those VPCs outside this resource by setting `manage_vpc_attachment` to false and using the **aviatrix_aws_tgw_vpc_attachment** resource instead of the in-line `attached_vpc {}` block.

» Example Usage

```
# Create an Aviatrix AWS TGW
resource "aviatrix_aws_tgw" "test_aws_tgw" {
  account_name                      = "devops"
  attached_aviatrix_transit_gateway = [
     "avx-transit-gw"
  ]
  aws_side_as_number                = "64512"
  manage_vpc_attachment             = true
  region                            = "us-east-1"
  tgw_name                          = "test-AWS-TGW"

  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 = "devops2"
    vpc_id           = "vpc-0e2fac2b91"
    vpc_region       = "us-east-1"
  }

  attached_vpc {
    vpc_account_name = "devops2"
    vpc_id           = "vpc-0c63660a16"
    vpc_region       = "us-east-1"
  }

  attached_vpc {
    vpc_account_name = "devops"
    vpc_id           = "vpc-032005cc444"
    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-03200566666"
      customized_routes                = "10.8.0.0/16,10.9.0.0/16"
      disable_local_route_propagation = true
    }
  }

  security_domains {
    security_domain_name = "firewall-domain"
    aviatrix_firewall    = true
  }
}
```

» Argument Reference

The following arguments are supported:

» Required

- `tgw_name` - (Required) Name of the AWS TGW to be created
- `account_name` - (Required) Name of the cloud account in the Aviatrix controller.
- `region` - (Required) AWS region of AWS TGW to be created in
- `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, along with the connections between them, are created automatically. These three domains can't be deleted, but the connection between any two of them can be.
  - `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 to be used as an Aviatrix Firewall Domain for the Aviatrix Firewall Network. Valid values: true, false. Default value: false.
  - `native_egress` - (Optional) Set to true if the security domain is to be used as a native egress domain (for non-Aviatrix Firewall Network-based central Internet bound traffic). Valid values: true, false. Default value: false.
  - `native_firewall` - (Optional) Set to true if the security domain is to be used as a native firewall domain (for non-Aviatrix Firewall Network-based firewall traffic inspection). 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.

» VPC Attachments

NOTE: The `attached_vpc` code block is to be nested under the `security_domains` block. Please see the code examples above for more information.

- `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) Cloud account name of the VPC in the Aviatrix controller.
  - `vpc_id` - (Required) VPC ID of the VPC to be attached to the security domain
  - `subnets` - (Optional) Advanced option. VPC subnets separated by ',' to attach to the VPC. If left blank, the Aviatrix Controller automatically selects a subnet representing each AZ for the VPC attachment. Example: "subnet-214f5646,subnet-085e8c81a89d70846".
  - `route_tables` - (Optional) Advanced option. Route tables separated by ',' to participate in TGW Orchestrator, i.e., learned routes will be propagated to these route tables. Example: "rtb-212ff547,rtb-045397874c170c745".
  - `customized_routes` - (Optional) Advanced option. Customized Spoke VPC Routes. It allows the admin to enter non-RFC1918 routes in the VPC route table targeting the TGW. Example: "10.8.0.0/16,10.9.0.0/16,10.10.0.0/16".
  - `customized_route_advertisement` - (Optional) Advanced option. Customized route(s) to be advertised to other VPCs that are connected to the same TGW. Example: "10.8.0.0/16,10.9.0.0/16,10.10.0.0/16".
  - `disable_local_route_propagation` - (Optional) Advanced option. If set to true, it disables automatic route propagation of this VPC to other VPCs within the same security domain. Valid values: true, false. Default value: false.

» Misc.

- `attached_aviatrix_transit_gateway` - (Optional) A list of names of Aviatrix Transit Gateway(s) (transit VPCs) to attach to the Aviatrix_Edge_Domain.
- `manage_transit_gateway_attachment` - (Optional) This parameter is a switch used to determine whether or not to manage transit gateway attachments to the TGW using the **aviatrix_aws_tgw** resource. If this is set to false, attachment of transit gateways must be done using the **aviatrix_aws_tgw_transit_gateway_attachment** resource. Valid

values: true, false. Default value: true.

- `manage_vpc_attachment` - (Optional) This parameter is a switch used to determine whether or not to manage VPC attachments to the TGW using the **aviatrix_aws_tgw** resource. If this is set to false, attachment of VPCs must be done using the **aviatrix_aws_tgw_vpc_attachment** resource. Valid values: true, false. Default value: true.

NOTE: `manage_vpc_attachment` - If you are using/upgraded to Aviatrix Terraform Provider R1.5+, and an **aviatrix_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.

NOTE: `manage_transit_gateway_attachment` - If you are using/upgraded to Aviatrix Terraform Provider R2.13+, and an **aviatrix_aws_tgw** resource was originally created with a provider version <R2.13, you must do 'terraform refresh' to update and apply the attribute's default value (true) into the state file.

» Import

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". **NOTE:** If `manage_transit_gateway_attachment` is set to "false", import action will also import the information of the transit gateway attached to TGW into the state file. Will need to do `terraform apply` to sync `manage_transit_gateway_attachment` to "false".

» aviatrix_aws_tgw_directconnect

The **aviatrix_aws_tgw_directconnect** resource allows the creation and management of Aviatrix AWS TGW DirectConnect connections.

» Example Usage

```
# Create an Aviatrix AWS TGW Directconnect
resource "aviatrix_aws_tgw_directconnect" "test_aws_tgw_directconnect" {
  tgw_name                   = "my-aws-tgw-1"
  directconnect_account_name = "username"
  dx_gateway_id              = "30321d76-dd01-49bf"
```

```
  security_domain_name       = "my-sdn-1"
  allowed_prefix             = "10.12.0.0/24"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `tgw_name` - (Required) This parameter represents the name of an AWS TGW.
- `directconnect_account_name` - (Required) This parameter represents the name of an Account in Aviatrix controller.
- `dx_gateway_id` - (Required) This parameter represents the name of a Direct Connect Gateway ID.
- `security_domain_name` - (Required) The name of a security domain, to which the direct connect gateway will be attached.
- `allowed_prefix` - (Required) A list of comma separated CIDRs for DXGW to advertise to remote(on-prem).
- `enable_learned_cidrs_approval` - (Optional) Switch to enable/disable encrypted transit approval for aws tgw direct connect. Valid values: true, false. Default value: false.

» Import

aws_tgw_directconnect can be imported using the `tgw_name` and `dx_gateway_id`, e.g.

```
$ terraform import aviatrix_aws_tgw_directconnect.test tgw_name~dx_gateway_id
```

» aviatrix_aws_tgw_transit_gateway_attachment

The **aviatrix_aws_tgw_transit_gateway_attachment** resource manages the attachment of the transit gateway to the AWS TGW.

» Example Usage

```
# Create an Aviatrix AWS TGW Transit Gateway Attachment
resource "aviatrix_aws_tgw_transit_gateway_attachment" "test_transit_gateway_attachment" {
  tgw_name             = "test-tgw"
```

```
  region               = "us-east-1"
  vpc_account_name     = "test-account"
  vpc_id               = "vpc-0e2fac2b91c6697b3"
  transit_gateway_name = "transit-gw-1"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `tgw_name` - (Required) Name of the AWS TGW.
- `region` - (Required) AWS Region of the TGW.
- `vpc_account_name` - (Required) The name of the cloud account in the Aviatrix controller, which is associated with the VPC.
- `vpc_id` - (Required) VPC ID of the VPC, where transit gateway is launched.
- `transit_gateway_name` - (Required) Name of the transit gateway to be attached to the AWS TGW.

» Import

aws_tgw_transit_gateway_attachment can be imported using the `tgw_name` and `vpc_id`, e.g.

```
$ terraform import aviatrix_aws_tgw_transit_gateway_attachment.test tgw_name~vpc_id
```

» aviatrix_aws_tgw_vpc_attachment

The **aviatrix_aws_tgw_vpc_attachment** resource manages the attaching & detaching of the VPC to & from an AWS TGW, and FireNet Gateway to TGW Firewall Domain.

» Example Usage

```
# Create an Aviatrix AWS TGW VPC Attachment
resource "aviatrix_aws_tgw_vpc_attachment" "test_aws_tgw_vpc_attachment" {
  tgw_name             = "test-tgw"
  region               = "us-east-1"
  security_domain_name = "my-sdn"
```

```
  vpc_account_name     = "test-account"
  vpc_id               = "vpc-0e2fac2b91c6697b3"
}
```

» Argument Reference

The following arguments are supported:

» Required

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

NOTE: If used to attach/detach FireNet Transit Gateway to/from TGW Firewall Domain, `vpc_id` is the ID of the Security VPC, and `security_domain_name` is the domain name of the Aviatrix Firewall Domain in TGW.

» Advanced Options

- `subnets` - (Optional and ForceNew) Advanced option. VPC subnets separated by ',' to attach to the VPC. If left blank, the Aviatrix Controller automatically selects a subnet representing each AZ for the VPC attachment. Example: "subnet-214f5646,subnet-085e8c81a89d70846".
- `route_tables` - (Optional and ForceNew) Advanced option. Route tables separated by ',' to participate in TGW Orchestrator, i.e., learned routes will be propagated to these route tables. Example: "rtb-212ff547,rtb-045397874c170c745".
- `customized_routes` - (Optional) Advanced option. Customized Spoke VPC Routes. It allows the admin to enter non-RFC1918 routes in the VPC route table targeting the TGW. Example: "10.8.0.0/16,10.9.0.0/16,10.10.0.0/16".
- `customized_route_advertisement` - (Optional and ForeNew) Advanced option. Customized route(s) to be advertised to other VPCs that are connected to the same TGW. Example: "10.8.0.0/16,10.9.0.0/16,10.10.0.0/16".
- `disable_local_route_propagation` - (Optional and ForceNew) Advanced option. If set to true, it disables automatic route propagation of

this VPC to other VPCs within the same security domain. Valid values: true, false. Default value: false.

» Import

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_ic
```

» 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          = "test-tgw1"
  route_domain_name = "Default_Domain"
  connection_name   = "my-conn1"
  connection_type   = "dynamic"
  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          = "test-tgw1"
  route_domain_name = "Default_Domain"
  connection_name   = "my-conn1"
  connection_type   = "static"
  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:

» Required

- `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.
- `connection_name` - (Required) Unique name of the connection.
- `public_ip` - (Required) Public IP address. Example: "40.0.0.0".
- `connection_type` - (Optional) Connection type. Valid values: 'dynamic', 'static'. 'dynamic' stands for a BGP VPN connection; 'static' stands for a static VPN connection. Default value: 'dynamic'.

NOTE: `connection_type` - If you are using/upgraded to Aviatrix Terraform Provider R2.11.0+, and an **aviatrix_aws_tgw_vpn_conn** resource (static VPN connection) was originally created with a provider version <R2.11.0, you must add `connection_type = static` into your configuration file and do 'terraform refresh' to update and apply the attribute's value (static) into the state file.

- `remote_as_number` - (Optional) AWS side as a number. Integer between 1-65535. Example: "12". **Required for a dynamic VPN connection.**
- `remote_cidr` - (Optional) Remote CIDRs separated by ",". Example: AWS: "16.0.0.0/16,16.1.0.0/16". **Required for a static VPN connection.**

» Optional

- `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.
- `enable_learned_cidrs_approval` - (Optional) Switch to enable/disable encrypted transit approval for aws tgw vpn connection. Valid values: true, false. Default value: false.

» 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

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_spoke_vpc

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

WARNING: The `aviatrix_spoke_vpc` resource is deprecated as of **Release 2.0**. It is currently kept for backward-compatibility and will be removed in the future. Please use the spoke gateway resource instead. If this is already in the state, please remove it from the state file and import as `aviatrix_spoke_gateway`.

» 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: ”CIDRZONE-SubnetName”, 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_vpc

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

WARNING: The `aviatrix_transit_vpc` resource is deprecated as of **Release 2.0**. It is currently kept for backward-compatibility and will be removed in the future. Please use the transit gateway resource instead. If this is already in the state, please remove it from state file and import as `aviatrix_transit_gateway`.

» 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: ”mygooglecloudvpc-name”, 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_spoke_gateway

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

» Example Usage

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

» Argument Reference

The following arguments are supported:

- `gw_name` - (Required) Spoke gateway name. This can be used for getting spoke 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:

- `gw_name` - Aviatrix spoke gateway name.
- `account_name` - Aviatrix account name.
- `cloud_type` - Type of cloud service provider.
- `vpc_id` - VPC ID.
- `vpc_reg` - VPC Region.
- `gw_size` - Instance type.
- `subnet` - Range of the subnet where the spoke gateway is launched.
- `public_ip` - Public IP address of the spoke gateway created.
- `allocate_new_eip` - Description: "Whether the eip is newly allocated or not.
- `single_az_ha`*Enable/Disable this feature.
- `transit_gw` - The transit gateway that the spoke gateway is attached to.

- `tag_list` - Instance tag of cloud provider. Only supported for AWS provider.
- `insane_mode` - Enable/Disable Insane Mode for Spoke Gateway.
- `insane_mode_az` - AZ of subnet being created for Insane Mode Spoke Gateway. Required if insane_mode is enabled for aws cloud.
- `enable_active_mesh` - Enable/Disable Active Mesh Mode for Spoke Gateway.
- `enable_vpc_dns_server` - Enable/Disalbe vpc_dns_server for Gateway.
- `enable_encrypt_volume` - Enable encrypt gateway EBS volume. Only supported for AWS provider.
- `customized_spoke_vpc_routes` - A list of comma separated CIDRs to be customized for the spoke VPC routes. When configured, it will replace all learned routes in VPC routing tables, including RFC1918 and non-RFC1918 CIDRs. It applies to this spoke gateway only.
- `filtered_spoke_vpc_routes` - A list of comma separated CIDRs to be filtered from the spoke VPC route table. When configured, filtering CIDR(s) or it's subnet will be deleted from VPC routing tables as well as from spoke gateway's routing table. It applies to this spoke gateway only.
- `included_advertised_spoke_routes` - A list of comma separated CIDRs to be advertised to on-prem as 'Included CIDR List'. When configured, it will replace all advertised routes from this VPC.
- `cloud_instance_id` - Cloud instance ID

» aviatrix_transit_gateway

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

» Example Usage

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

» Argument Reference

The following arguments are supported:

- `gw_name` - (Required) Transit gateway name. This can be used for getting transit 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:

- `gw_name` - Aviatrix transit gateway name.
- `account_name` - Aviatrix account name.
- `cloud_type` - Type of cloud service provider.
- `vpc_id` - VPC ID.
- `vpc_reg` - VPC Region.
- `gw_size` - Instance type.
- `subnet` - Range of the subnet where the transit gateway is launched.
- `public_ip` - Public IP address of the Gateway created.
- `allocate_new_eip` - Whether the eip is newly allocated or not.
- `single_az_ha` - Enable/Disable this feature.
- `tag_list` - Instance tag of cloud provider. Only supported for AWS provider.
- `enable_hybrid_connection` - Sign of readiness for TGW connection.
- `connected_transit` - Connected Transit status.
- `insane_mode` - Enable/Disable Insane Mode for Spoke Gateway.
- `insane_mode_az` - AZ of subnet being created for Insane Mode Spoke Gateway. Required if insane_mode is enabled for aws cloud.
- `enable_firenet` - Whether firenet interfaces is enabled.
- `enable_active_mesh` - Enable/Disable active mesh mode for Transit Gateway.
- `enable_vpc_dns_server` - Enable/Disable vpc_dns_server for Gateway. Only supports AWS.
- `enable_advertise_transit_cidr` - Enable/Disable advertise transit VPC network CIDR.
- `bgp_manual_spoke_advertise_cidrs` - Intended CIDR list to advertise to VGW.
- `enable_encrypt_volume` - Enable/Disable encrypt gateway EBS volume. Only supported for AWS provider.
- `customized_spoke_vpc_routes` - A list of comma separated CIDRs to be customized for the spoke VPC routes. When configured, it will replace all learned routes in VPC routing tables, including RFC1918 and non-RFC1918 CIDRs. It applies to all spoke gateways attached to this transit gateway.
- `filtered_spoke_vpc_routes` - A list of comma separated CIDRs to be filtered from the spoke VPC route table. When configured, filtering CIDR(s) or it's subnet will be deleted from VPC routing tables as well as from spoke gateway's routing table. It applies to all spoke gateways attached to this transit gateway.

- `excluded_advertised_spoke_routes` - A list of comma separated CIDRs to be advertised to on-prem as 'Excluded CIDR List'. When configured, it inspects all the advertised CIDRs from its spoke gateways and remove those included in the 'Excluded CIDR List'.

» aviatrix_spoke_gateway

The **aviatrix_spoke_gateway** resource allows the creation and management of 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 Azure Spoke Gateway
resource "aviatrix_spoke_gateway" "test_spoke_gateway_azure" {
  cloud_type   = 8
  account_name = "my-azure"
```

```
  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:

» Required

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1), GCP(4), AZURE(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. Example: AWS: "vpc-abcd1234", GCP: "vpc-gcp-test", AZURE: "vnet1:hello", OCI: "vpc-oracle-test1".
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", GCP: "us-west2-a", AZURE: "East US 2", Oracle: "us-ashburn-1".
- `gw_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", AZURE: "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". **NOTE: If using `insane_mode`, please see notes here.**

» HA

- `single_az_ha` (Optional) Set to true if this feature is desired. Valid values: true, false.
- `ha_subnet` - (Optional) HA Subnet. Required only if enabling HA for AWS/Azure gateway. Setting to empty/unsetting 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 only if enabling HA for GCP gateway. Setting to empty/unsetting 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_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 be allocated. Only available for AWS.
- `ha_gw_size` - (Optional) HA Gateway Size. Mandatory if enabling HA. Example: "t2.micro".

» Insane Mode

- `insane_mode` - (Optional) Enable Insane Mode for Spoke Gateway. Insane Mode gateway size has to be at least c5 (AWS) or Standard_D3_v2 (AZURE). If enabled, you must specify a valid /26 CIDR segment of the VPC to create a new subnet. Only supported for AWS and Azure. Valid values: true, false.
- `insane_mode_az` - (Optional) AZ of subnet being created for Insane Mode Spoke Gateway. Required for AWS if `insane_mode` is enabled. Example: AWS: "us-west-1a".

» SNAT/DNAT

- `single_ip_snat` - (Optional) Specify whether to enable Source NAT feature in "single_ip" mode on the gateway or not. Please disable AWS NAT instance before enabling this feature. Currently only supports AWS(1) and AZURE(8). Valid values: true, false.

NOTE: `enable_snat` has been renamed to `single_ip_snat` in provider version R2.10. Please see notes here for more information.

NOTE: Custom SNAT and DNAT support have been deprecated and functionality has been moved to **aviatrix_gateway_snat** and **aviatrix_gateway_dnat** respectively, in provider version R2.10. Please see notes for `snat_mode`, `snat_policy` and `dnat_policy` in the Notes section below.

» Encryption

- `enable_encrypt_volume` - (Optional) Enable EBS volume encryption for Gateway. Only supports AWS. Valid values: true, false. Default value: false.
- `customer_managed_keys` - (Optional and Sensitive) Customer managed key ID.

» Route Customization

- `customized_spoke_vpc_routes` - (Optional) A list of comma separated CIDRs to be customized for the spoke VPC routes. When configured, it will replace all learned routes in VPC routing tables, including RFC1918 and non-RFC1918 CIDRs. It applies to this spoke gateway only. Example: "10.0.0.0/116,10.2.0.0/16".
- `filtered_spoke_vpc_routes` - (Optional) A list of comma separated CIDRs to be filtered from the spoke VPC route table. When configured, filtering CIDR(s) or it's subnet will be deleted from VPC routing tables as well as from spoke gateway's routing table. It applies to this spoke gateway only. Example: "10.2.0.0/116,10.3.0.0/16".
- `included_advertised_spoke_routes` - (Optional) A list of comma separated CIDRs to be advertised to on-prem as 'Included CIDR List'. When configured, it will replace all advertised routes from this VPC. Example: "10.4.0.0/116,10.5.0.0/16".

» Misc.

- `transit_gw` - (Optional) Specify the Aviatrix transit gateway to attach this spoke gateway to.
- `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+. Valid values: true, false. Default: true. Option not available for GCP, AZURE and OCI gateways, they will automatically allocate new EIPs.
- `eip` - (Optional) Required when `allocate_new_eip` is false. It uses the specified EIP for this gateway. Available in Controller 4.7+. Only available for AWS.
- `tag_list` - (Optional) Instance tag of cloud provider. Only AWS, cloud_type is "1", is supported. Example: ["key1:value1", "key2:value2"].
- `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.
- `security_group_id` - Security group used for the spoke gateway.
- `cloud_instance_id` - Cloud instance ID of the spoke gateway.
- `private_ip` - Private IP address of the spoke gateway created.
- `ha_cloud_instance_id` - Cloud instance ID of the HA spoke gateway.

The following arguments are deprecated:

- `enable_snat` - (Optional) Specify whether enabling Source NAT feature on the gateway or not. Please disable AWS NAT instance before enabling this feature. Currently only supports AWS(1) and AZURE(8). Valid values: true, false.
- `snat_mode` - (Optional) Valid values: "primary", "secondary" and "custom". Default value: "primary".
- `snat_policy` - (Optional) Policy rule applied for "snat_mode" of "custom".
  - `src_ip` - (Optional) A source IP address range where the policy rule applies.
  - `src_port` - (Optional) A source port that the policy rule applies.
  - `dst_ip` - (Optional) A destination IP address range where the policy rule applies.
  - `dst_port` - (Optional) A destination port where the policy rule applies.
  - `protocol` - (Optional) A destination port protocol where the policy rule applies.
  - `interface` - (Optional) An output interface where the policy rule applies.
  - `connection` - (Optional) Default value: "None".
  - `mark` - (Optional) A tag or mark of a TCP session where the policy rule applies.
  - `new_src_ip` - (Optional) The changed source IP address when all specified qualifier conditions meet. One of the rule fields must be specified for this rule to take effect.
  - `new_src_port` - (Optional) The translated destination port when all specified qualifier conditions meet. One of the rule field must be specified for this rule to take effect.
  - `exclude_rtb` - (Optional) This field specifies which VPC private route table will not be programmed with the default route entry.
- `dnat_policy` - (Optional) Policy rule applied for enabling Destination NAT (DNAT), which allows you to change the destination to a virtual address range. Currently only supports AWS(1) and AZURE(8).
  - `src_ip` - (Optional) A source IP address range where the policy rule

applies.
- `src_port` - (Optional) A source port that the policy rule applies.
- `dst_ip` - (Optional) A destination IP address range where the policy rule applies.
- `dst_port` - (Optional) A destination port where the policy rule applies.
- `protocol` - (Optional) A destination port protocol where the policy rule applies.
- `interface` - (Optional) An output interface where the policy rule applies.
- `connection` - (Optional) Default value: "None".
- `mark` - (Optional) A tag or mark of a TCP session where the policy rule applies.
- `new_src_ip` - (Optional) The changed source IP address when all specified qualifier conditions meet. One of the rule fields must be specified for this rule to take effect.
- `new_src_port` - (Optional) The translated destination port when all specified qualifier conditions meet. One of the rule field must be specified for this rule to take effect.
- `exclude_rtb` - (Optional) This field specifies which VPC private route table will not be programmed with the default route entry.

» Import

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

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

» Notes

» insane_mode

If `insane_mode` is enabled, you must specify a valid /26 CIDR segment of the VPC specified for the `subnet`. This will then create a new subnet to be used for the corresponding gateway. You cannot specify an existing /26 subnet.

» enable_snat

If you are using/upgraded to Aviatrix Terraform Provider R2.10+, and a spoke gateway with `enable_snat` set to true was originally created with a provider version <R2.10, you must do a 'terraform refresh' to update and apply the attribute's value into the state. In addition, you must also change this attribute to `single_ip_snat` in your `.tf` file.

» snat_mode & snat_policy

If you are using/upgraded to Aviatrix Terraform Provider R2.10+, and a spoke gateway with `snat_mode` and `snat_policy` was originally created with a provider version <R2.10, you must do a 'terraform refresh' to remove attribute's value from the state. In addition, you must transfer its corresponding values to the **aviatrix_gateway_snat** resource in your `.tf` file and perform a 'terraform import' to rectify the state file.

» dnat_policy

If you are using/upgraded to Aviatrix Terraform Provider R2.10+, and a spoke gateway with `dnat_policy` was originally created with a provider version <R2.10, you must do a 'terraform refresh' to remove attribute's value from the state. In addition, you must its value to its corresponding **aviatrix_gateway_dnat** resource in your `.tf` file and perform a 'terraform import' to rectify the state file.

» aviatrix_transit_gateway

The **aviatrix_transit_gateway** resource allows the creation and management of 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 Azure 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:

» Required

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1), GCP(4), AZURE(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. Example: AWS: "vpc-abcd1234", GCP: "vpc-gcp-test", AZURE: "vnet1:hello", OCI: "vpc-oracle-test1".
- `vpc_reg` - (Required) Region of cloud provider. Example: AWS: "us-east-1", GCP: "us-west2-a", AZURE: "East US 2", Oracle: "us-ashburn-1".
- `gw_size` - (Required) Size of the gateway instance. Example: AWS: "t2.large", AZURE: "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". **NOTE: If using `insane_mode`, please see notes here.**

» HA

- `single_az_ha` (Optional) Set to true if this feature is desired. Valid values: true, false.
- `ha_subnet` - (Optional) HA Subnet CIDR. Required only if enabling HA for AWS/Azure gateway. Setting to empty/unsetting 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 only if enabling HA for GCP gateway. Setting to empty/unsetting 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 Transit HA Gateway. Required for AWS if `insane_mode` is enabled and `ha_subnet` is set. Example: AWS: "us-west-1a".
- `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 be allocated. Only available for AWS.
- `ha_gw_size` - (Optional) HA Gateway Size. Mandatory if enabling HA. Example: "t2.micro".

» Insane Mode

- `insane_mode` - (Optional) Specify Insane Mode high performance gateway. Insane Mode gateway size must be at least c5 size (AWS) or Stan-

dard_D3_v2 (AZURE). If enabled, you must specify a valid /26 CIDR segment of the VPC to create a new subnet. Only available for AWS and Azure. Valid 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".

» SNAT

- `single_ip_snat` - (Optional) Enable "single_ip" mode Source NAT for this container. Valid values: true, false. **NOTE: Please see notes here in regards to changes to this argument in R2.10.**

» Advanced Config

- `connected_transit` - (Optional) Specify Connected Transit status. If enabled, it allows spokes to run traffics to other spokes via transit gateway. 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. **NOTE: If previously enabled through vgw_conn resource prior to provider version R2.6, please see notes here.**
- `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. **NOTE: If previously enabled through vgw_conn resource prior to provider version R2.6, please see notes here.**
- `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. **NOTE: If previously using an older provider version R2.5 where attribute name was `enable_firenet_interfaces`, please see notes here.**

NOTE: Enabling FireNet will automatically enable hybrid connection. If `enable_firenet` is set to true, please set `enable_hybrid_connection` to true in the respective **aviatrix_transit_gateway** as well.

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

» Encryption

- `enable_encrypt_volume` - (Optional) Enable EBS volume encryption for Gateway. Only supports AWS. Valid values: true, false. Default value: false.

- `customer_managed_keys` - (Optional and Sensitive) Customer managed key ID.

» Route Customization

- `customized_spoke_vpc_routes` - (Optional) A list of comma separated CIDRs to be customized for the spoke VPC routes. When configured, it will replace all learned routes in VPC routing tables, including RFC1918 and non-RFC1918 CIDRs. It applies to all spoke gateways attached to this transit gateway. Example: "10.0.0.0/116,10.2.0.0/16".
- `filtered_spoke_vpc_routes` - (Optional) A list of comma separated CIDRs to be filtered from the spoke VPC route table. When configured, filtering CIDR(s) or it's subnet will be deleted from VPC routing tables as well as from spoke gateway's routing table. It applies to all spoke gateways attached to this transit gateway. Example: "10.2.0.0/116,10.3.0.0/16".
- `excluded_advertised_spoke_routes` - (Optional) A list of comma separated CIDRs to be advertised to on-prem as 'Excluded CIDR List'. When configured, it inspects all the advertised CIDRs from its spoke gateways and remove those included in the 'Excluded CIDR List'. Example: "10.4.0.0/116,10.5.0.0/16".

» Misc.

- `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+. Valid values: true, false. Default: true. Option not available for GCP, Azure and OCI gateways, they will automatically allocate new EIPs.
- `eip` - (Optional) Required when `allocate_new_eip` is false. It uses the specified EIP for this gateway. Available in Controller version 4.7+. Only available for AWS.
- `tag_list` - (Optional) Instance tag of cloud provider. Only supported for AWS. Example: ["key1:value1","key2:value2"].
- `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_learned_cidrs_approval` - (Optional) Switch to enable/disable encrypted transit approval for transit Gateway. 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.
- `security_group_id` - Security group used for the transit gateway.
- `cloud_instance_id` - Cloud instance ID of the transit gateway.
- `private_ip` - Private IP address of the transit gateway created.
- `ha_cloud_instance_id` - Cloud instance ID of the HA transit gateway.

The following arguments are deprecated:

- `enable_firenet_interfaces` - (Optional) Sign of readiness for FireNet connection. Valid values: true, false. Default value: false.
- `enable_snat` - (Optional) Enable Source NAT for this container. Valid values: true, false.

» Import

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

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

» Notes

» CIDR advertising

`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.

» 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.

» insane_mode

If `insane_mode` is enabled, you must specify a valid /26 CIDR segment of the VPC specified for the `subnet`. This will then create a new subnet to be used for the corresponding gateway. You cannot specify an existing /26 subnet.

» enable_snat

If you are using/upgraded to Aviatrix Terraform Provider R2.10+, and a transit gateway with `enable_snat` set to true was originally created with a provider version <R2.10, you must do a 'terraform refresh' to update and apply the attribute's value into the state. In addition, you must also change this attribute to `single_ip_snat` in your `.tf` file.

» aviatrix_transit_gateway_peering

The **aviatrix_transit_gateway_peering** resource allows the creation and management of peerings between Aviatrix transit gateways.

» Example Usage

```
# Create an Aviatrix Transit Gateway Peering
resource "aviatrix_transit_gateway_peering" "test_transit_gateway_peering" {
  transit_gateway_name1 = "transit-Gw1"
  transit_gateway_name2 = "transit-Gw2"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `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

transit_gateway_peering 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_gatew
```

» aviatrix_vgw_conn

The **aviatrix_vgw_conn** resource manages the connection between the Aviatrix transit gateway and AWS VGW.

» 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:

» Required

- `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 VGW that will be used for this connection. Example: "vgw-abcd1234".
- `bgp_vgw_account` - (Required) Cloud Account used to create the AWS VGW that will be used for this connection. Example: "dev-account-1".

- `bgp_vgw_region` - (Required) Region of AWS VGW that will be 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

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_azure_spoke_native_peering

The **aviatrix_azure_spoke_native_peering** resource allows the creation and management of Aviatrix Azure Spoke VNet attachments via Native Peering.

» Example Usage

```
# Create an Aviatrix Azure spoke native peering
resource "aviatrix_azure_spoke_native_peering" "test" {
  transit_gateway_name = "transit-gw-azure"
  spoke_account_name   = "devops-azure"
  spoke_region         = "West US"
  spoke_vpc_id         = "Foo_VNet:Bar_RG"
```

```
}
```

» Argument Reference

The following arguments are supported:

» Required

- `transit_gateway_name` - (Required) Name of an Transit FireNet-enabled Azure transit gateway.
- `spoke_account_name` - (Required) An Aviatrix account that corresponds to a subscription in Azure.
- `spoke_region` - (Required) Spoke VNet region. Example: "West US".
- `spoke_vpc_id` - (Required) Combination of the Spoke's VNet name and resource group. Example: "Foo_VNet:Bar_RG".

» Import

azure_spoke_native_peering can be imported using the `transit_gateway_name`, `spoke_account_name` and `spoke_vpc_id`, e.g.

```
$ terraform import aviatrix_azure_spoke_native_peering.test transit_gateway_name~spoke_accou
```

» aviatrix_transit_firenet_policy

The **aviatrix_transit_firenet_policy** resource allows the creation and management of Aviatrix Transit FireNet policies that determine which resources should be inspected in the Transit FireNet solution.

» Example Usage

```
# Create an Aviatrix Transit FireNet Policy
resource "aviatrix_transit_firenet_policy" "test_transit_firenet_policy" {
  transit_firenet_gateway_name = "transitGw1"
  inspected_resource_name      = "SPOKE:spokeGw1"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `transit_firenet_gateway_name` - (Required) Name of the Transit FireNet-enabled transit gateway. Currently supports AWS and Azure.
- `inspected_resource_name` - (Required) The name of the resource which will be inspected.

» Import

transit_firenet_policy can be imported using the `transit_firenet_gateway_name` and `inspected_resource_name`, e.g.

```
$ terraform import aviatrix_transit_firenet_policy.test transit_firenet_gateway_name~inspect
```

» aviatrix_firewall_management_access

The **aviatrix_firewall_management_access** resource allows the management of which resource to permit visibility into the Transit (FireNet) VPC.

» Example Usage

```
# Create an Aviatrix Firewall Management Access
resource "aviatrix_firewall_management_access" "test_firewall_management_access" {
  transit_firenet_gateway_name    = "transit-gw"
  management_access_resource_name = "SPOKE:spoke-gw"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `transit_firenet_gateway_name` - (Required) Name of the Transit FireNet-enabled transit gateway. Currently supports AWS(1) and Azure(8) providers.
- `management_access_resource_name` - (Required) Name of the resource to enable Firewall Management Access.

» Import

firewall_management_access can be imported using the `transit_firenet_gateway_name`, e.g.

```
$ terraform import aviatrix_firewall_management_access.test transit_firenet_gateway_name
```

» aviatrix_firenet

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

» Example Usage

```
# Aviatrix FireNet Data Source
data "aviatrix_firenet" "foo" {
  vpc_id = "vpc-abcdef"
}
```

» Argument Reference

The following arguments are supported:

- `vpc_id` - (Required) ID of the Security VPC.

» Attribute Reference

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

- `vpc_id` - ID of the Security VPC.
- `inspection_enabled` - Enable/Disable traffic inspection.
- `egress_enabled` - Enable/Disable egress through firewall.
- `firewall_instance_association` - List of firewall instances associated with fireNet.
  - `firenet_gw_name` - Name of the primary FireNet gateway.
  - `instance_id` - ID of Firewall instance.
  - `vendor_type` - Type of the firewall.
  - `firewall_name` - Firewall instance name.
  - `lan_interface`- Lan interface ID.
  - `management_interface` - Management interface ID.
  - `egress_interface`- Egress interface ID.
  - `attached`- Switch to attach/detach firewall instance to/from fireNet.

» aviatrix_firenet_vendor_integration

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

NOTE: FireNet with Panorama should be set up using the **aviatrix_firenet_firewall_manager** data source. Do not use `save` or `sync` options listed below.

NOTE: aviatrix_firenet_firewall_manager is currently under development.

» 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 Networks 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 Networks VM-Series", "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.
- `number_of_retries` - (Optional) Number of retries for `save` or `synchronize`. Example: 1. Default value: 0.

- `retry_interval` - (Optional) Retry interval in seconds for `save` or `synchronize`. Example: 120. Default value: 300.
- `save` - (Optional) Switch to save or not.
- `synchronize` - (Optional) Switch to sync or not.

» aviatrix_firenet

The **aviatrix_firenet** resource allows the creation and management of Aviatrix FireNets (Firewall Networks).

NOTE: This resource is used in conjunction with multiple other resources that may include, and are not limited to: **firewall_instance**, **aws_tgw**, and **transit_gateway** resources, under the Aviatrix FireNet solution. Explicit dependencies may be set using `depends_on`. For more information on proper FireNet configuration, please see the workflow here.

» 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 {
    firenet_gw_name = "avx-firenet-gw"
    instance_id     = "avx-fqdn-gateway"
```

```
    vendor_type     = "fqdn_gateway"
    attached        = true
  }
}
```

» Argument Reference

The following arguments are supported:

» Required

- `vpc_id` - (Required) VPC ID of the Security VPC.
- `inspection_enabled` - (Optional) Enable/disable traffic inspection. Valid values: true, false. Default value: true.

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

- `egress_enabled` - (Optional) Enable/disable egress through firewall. Valid values: true, false. Default value: false.

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

» Firewall Association

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

- `firewall_instance_association` - (Optional) Dynamic block of firewall instance(s) to be associated with the FireNet.
  - `firenet_gw_name` - (Required) Name of the primary FireNet gateway.
  - `instance_id` - (Required) ID of Firewall instance. If associating FQDN gateway to FireNet, it is FQDN gateway's `gw_name`.
  - `vendor_type` - (Optional) Type of firewall. Valid values: "Generic", "fqdn_gateway". Default value: "Generic". Value "fqdn_gateway" is required for FQDN gateway.
  - `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.

» Import

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

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

» aviatrix_firewall_instance

The **aviatrix_firewall_instance** resource allows the creation and management 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:

» Required

- `vpc_id` - (Required) VPC 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/Azure 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".
- `firewall_image_version` - (Optional) Version of firewall image. If not specified, Controller will automatically select the latest version available.

» Advanced Options

- `key_name`- (Optional) The **.pem** filename 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.
- `username`- (Optional) Applicable to Azure deployment only. "admin" as a username is not accepted.
- `key_name`- (Optional) Applicable to Azure deployment only.

» 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

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

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

» aviatrix_arm_peer

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

WARNING: The `aviatrix_arm_peer` resource is deprecated as of **Release 2.12**. It is currently kept for backward-compatibility and will be removed in the future. Please use the Azure peer resource instead. If this is already in the state, please remove it from the state file and import as `aviatrix_azure_peer`.

» 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:

» Required

- `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

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_group
```

» aviatrix_azure_peer

The **aviatrix_azure_peer** resource allows the creation and management of the Aviatrix peerings between Azure VNets.

» Example Usage

```
# Create an Aviatrix Azure Peering
resource "aviatrix_azure_peer" "test_azurepeer" {
  account_name1             = "test1-account"
  account_name2             = "test2-account"
  vnet_name_resource_group1 = "Foo_VNet1:Bar_RG1"
  vnet_name_resource_group2 = "Foo_VNet2:Bar_RG2"
  vnet_reg1                 = "Central US"
  vnet_reg2                 = "East US"
}
```

» Argument Reference

The following arguments are supported:

» Required

- account_name1 - (Required) Name of the Azure cloud account in the Aviatrix controller for VNet 1.
- account_name2 - (Required) Name of the Azure cloud account in the Aviatrix controller for VNet 2.
- vnet_name_resource_group1 - (Required) Azure VNet 1's name. Example: "VNet_Name:Resource_Group_Name".
- vnet_name_resource_group2 - (Required) Azure VNet 2's name. Example: "VNet_Name:Resource_Group_Name".

- `vnet_reg1` - (Required) Region of Azure VNet 1. Example: "East US 2".
- `vnet_reg2` - (Required) Region of Azure VNet 2. 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

azure_peer can be imported using the `vnet_name_resource_group1` and `vnet_name_resource_group2`, e.g.

```
$ terraform import aviatrix_azure_peer.test vnet_name_resource_group1~vnet_name_resource_gro
```

» 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:

» Required

- `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

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_trans_peer

The **aviatrix_trans_peer** resource allows the creation and management of Aviatrix Encrypted Transitive Peering.

» Example Usage

```
# Create an Aviatrix AWS Transitive Peering
resource "aviatrix_trans_peer" "test_trans_peer" {
  source         = "avtx-us-east-gw1"
  nexthop        = "avtx-us-east-gw2"
  reachable_cidr = "10.152.0.0/16"
}
```

» Argument Reference

The following arguments are supported:

» Required

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

» Import

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 Encrypted Peering tunnels).

» Example Usage

```
# Create an Aviatrix AWS Tunnel
resource "aviatrix_tunnel" "test_tunnel" {
  gw_name1 = "avtx-gw1"
  gw_name2 = "avtx-gw2"
}
```

» Argument Reference

The following arguments are supported:

» Required

- `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.

» HA

- `enable_ha` - (Optional) Enable this attribute if peering-HA is enabled on the gateways. Valid values: true, false. Default value: 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

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

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

» aviatrix_site2cloud

The **aviatrix_site2cloud** resource creates and manages Aviatrix Site2Cloud connections.

» Example Usage

```
# Create an Aviatrix Site2cloud Connection
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:

» Required

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

» HA

- `ha_enabled` - (Optional) Specify whether or not to enable HA. Valid Values: true, false. **NOTE: Please see notes here regarding HA requirements.**
- `backup_gateway_name` - (Optional) Backup gateway name. **NOTE: Please see notes here regarding HA requirements.**
- `backup_remote_gateway_ip` - (Optional) Backup Remote Gateway IP. **NOTE: Please see notes here regarding HA requirements.**

- `backup_pre_shared_key` - (Optional) Backup Pre-Shared Key.

» Custom Algorithms

- `custom_algorithms` - (Optional) Switch to enable custom/non-default algorithms for IPSec Authentication/Encryption. Valid values: true, false. **NOTE: Only supported for 'udp' tunnel type. Please see notes here for more information.**
- `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'.

» Encryption over ExpressRoute/DirectConnect

- `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.

» Misc.

- `pre_shared_key` - (Optional) Pre-Shared Key. Only available for "udp" tunnel_type.

- `ssl_server_pool` - (Optional) Specify ssl_server_pool for tunnel_type "tcp". Default value: "192.168.44.0/24". **NOTE: Only supported for 'tcp' tunnel type. Please see notes here for more information.**
- `enable_dead_peer_detection` - (Optional) Enable/disable Deed Peer Detection for an existing site2cloud connection. Default value: true. **NOTE: Please see notes here in regards to any deltas found in your state with the addition of this argument in R1.9**
- `enable_active_active` - (Optional) Enable/disable active active HA for an existing site2cloud connection. Valid values: true, false. Default value: false.

» Attribute Reference

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

- `local_subnet_cidr` - Local subnet CIDR.

» Import

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

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

» Notes

» 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.

» enable_dead_peer_detection

If you are using/upgraded to Aviatrix Terraform Provider R1.9+, and a 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.

» HA Enabled

The following arguments are only supported if the backup gateway is set up by enabling peering HA through the primary gateway resource by specifying

a `peering_ha_subnet` and `peering_ha_gw_size`. For more information on site2cloud, please see the doc site here:

- `backup_gateway_name`
- `backup_remote_gateway_ip`
- `ha_enabled`

» 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 the default value.

» aviatrix_geo_vpn

The **aviatrix_geo_vpn** resource enables and manages the Aviatrix Geo VPN.

» Example Usage

```
# Create an Aviatrix Geo VPN
resource "aviatrix_geo_vpn" "test_geo_vpn" {
  cloud_type    = 1
  account_name  = "devops-aws"
  service_name  = "vpn"
  domain_name   = "aviatrix.live"
  elb_dns_names = [
    "elb-test1-497f5e89.elb.us-west-1.amazonaws.com",
    "elb-test2-974f895e.elb.us-east-2.amazonaws.com",
  ]
}
```

» Argument Reference

The following arguments are supported:

» Required

- `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.

- `domain_name` - (Required) The hosted domain name. It must be hosted by AWS Route53 or Azure DNS in the selected account.
- `service_name` - (Required) The hostname that users will connect to. A DNS record will be created for this name in the specified domain name.
- `elb_dns_names` - (Required) List of ELB names to attach to this Geo VPN name.

» Import

geo_vpn can be imported using the `service_name` and `domain_name`, e.g.

```
$ terraform import aviatrix_geo_vpn.test service_name~domain_name
```

» aviatrix_saml_endpoint

The **aviatrix_saml_endpoint** 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:

» Required

- `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

- `custom_entity_id` - (Optional) Custom Entity ID. Required to be non-empty for 'Custom' Entity ID type, empty for 'Hostname' Entity ID type.
- `custom_saml_request_template` - (Optional) Custom SAML Request Template in string.

» Import

saml_endpoint can be imported using the SAML `endpoint_name`, e.g.

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

» aviatrix_vpn_profile

The **aviatrix_vpn_profile** resource allows the creation and management of Aviatrix 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:

» Required

- `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 whitelist or blacklist.

» Policy Options

- `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

vpn_profile can be imported using the VPN profile's `name`, e.g.

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

» aviatrix_vpn_user

The **aviatrix_vpn_user** resource creates and manages Aviatrix 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:

» Required

- `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".

» SAML

- `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

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

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

» aviatrix_vpn_user_accelerator

The **aviatrix_vpn_user_accelerator** resource manages the Aviatrix VPN User Accelerator.

» 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:

» Required

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

» Import

vpn_user_accelerator can be imported using the `elb_name`, e.g.

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

» aviatrix_firewall

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

» Example Usage

```
# Create an Aviatrix Firewall
resource "aviatrix_firewall" "stateful_firewall_1" {
  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      = "allow"
    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"
  }

  policy {
    protocol    = "tcp"
    src_ip      = "10.15.2.224/32"
    log_enabled = false
    dst_ip      = "10.12.3.172/32"
    action      = "force-drop"
    port        = "0:65535"
    description = "This is policy no.3"
  }
}
```

» Argument Reference

The following arguments are supported:

- `gw_name` - (Required) Gateway name to attach firewall policy to.
- `base_policy` - (Optional) New base policy. Valid Values: "allow-all", "deny-all". Default value: "deny-all"
- `base_log_enabled` - (Optional) Indicates whether enable logging or not. Valid Values: true, false. Default value: 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" and "force-drop" (in stateful firewall rule to allow immediate packet dropping on established sessions).
  - `log_enabled`- (Optional) Valid values: true, false. Default value:

false.
  - `description`- (Optional) Description of the policy. Example: "This is policy no.1".

» Import

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

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

» aviatrix_firewall_tag

The **aviatrix_firewall_tag** resource allows the creation and management of Aviatrix Stateful Firewall tags for tag-based security for gateways.

» 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:

» Required

- `firewall_tag` - (Required) Name of the stateful firewall tag to be created.

» Tag Rules

- `cidr_list` - (Optional) Dynamic block representing a CIDR to filter, and a name to identify it:
  - `cidr_tag_name` - (Required) A name to identify the CIDR. Example: "policy1".
  - `cidr` - (Required) CIDR address to filter. Example: "10.88.88.88/32".

» Import

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 gateways.

» 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        = "test-gw1"
    source_ip_list = [
      "172.31.0.0/16",
      "172.31.0.0/20"
    ]
  }

  gw_filter_tag_list {
    gw_name        = "test-gw2"
    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 FQDN mode: whitelist or blacklist. 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, `single_ip_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 `single_ip_snat = true` in their gateway resource.

NOTE: In order for the FQDN feature to be enabled, the corresponding

gateway's `enable_vpc_dns_server` must be set to `false` at creation. FQDN will automatically enable that feature, which will cause a diff in the state. Please add `lifecycle { ignore_changes = [enable_vpc_dns_server] }` within that gateway's resource block in order to workaround this known issue. See here for more information about the `lifecycle` attribute in Terraform.

» Import

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

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

» aviatrix_vpc

The **aviatrix_vpc** resource allows the creation and management of VPCs of various cloud types.

» Example Usage

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

# Create a GCP VPC
resource "aviatrix_vpc" "gcp_vpc" {
  cloud_type           = 4
  account_name         = "devops"
  name                 = "gcp-vpc"

  subnets {
    name   = "subnet-1"
    region = "us-west1"
    cidr   = "10.10.0.0/24"
  }

  subnets {
```

```
    name   = "subnet-2"
    region = "us-west2"
    cidr   = "10.11.0.0/24"
  }
}

# Create an Azure VNet
resource "aviatrix_vpc" "azure_vnet" {
  cloud_type           = 8
  account_name         = "devops"
  region               = "Central US"
  name                 = "azure-vnet"
  cidr                 = "12.0.0.0/16"
  aviatrix_firenet_vpc = false
}
```

» Argument Reference

The following arguments are supported:

» Required

- `cloud_type` - (Required) Type of cloud service provider, requires an integer value. Currently only AWS(1), GCP(4) and AZURE(8) are 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` - (Optional) Region of cloud provider. **Required to be empty for GCP provider, and non-empty for other providers.** Example: AWS: "us-east-1", AZURE: "East US 2".
- `cidr` - (Optional) VPC CIDR. **Required to be empty for GCP provider, and non-empty for other providers.** Example: "10.11.0.0/24".

» Google Cloud

- `subnets` - (Optional) List of subnets to be specify for GCP provider. Required to be non-empty for GCP provider, and empty for other providers.
  - `region` - Region of this subnet.
  - `cidr` - CIDR block.
  - `name` - Name of this subnet.

» Misc.

- `aviatrix_transit_vpc` - (Optional) Specify whether it is an Aviatrix Transit VPC to be used for Transit Network or TGW solutions. **Only AWS is supported. Required to be false for other providers.** Valid values: true, false. Default: false.
- `aviatrix_firenet_vpc` - (Optional) Specify whether it is an Aviatrix FireNet VPC to be used for Aviatrix FireNet and Transit FireNet solutions. **Only AWS and Azure are supported. Required to be false for other providers.** 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.

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

vpc can be imported using the VPC's `name`, e.g.

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

» 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_controller_config

The **aviatrix_controller_config** resource allows management of an Aviatrix Controller's configurations.

» 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"
}
```

```
# Create an Aviatrix Controller Config with Cloudn Backup Configuration Enabled
resource "aviatrix_controller_config" "test_controller_config" {
  backup_configuration = true
  backup_cloud_type    = 1
  backup_account_name  = "account_example"
  backup_bucket_name   = "bucket_example"
}
```

» Argument Reference

The following arguments are supported:

» Security Options

- `sg_management_account_name` - (Optional) Select the primary access account.
- `security_group_management` - (Optional) Enable to allow Controller to automatically manage inbound rules from gateways. Valid values: true, false. Default value: false.
- `http_access` - (Optional) Switch for HTTP access. Valid values: true, false. Default value: false.
- `fqdn_exception_rule` - (Optional) Enable/disable packets without an SNI field to pass through gateway(s). Valid values: true, false. Default value: true. For more information on this setting, please see here

» Backup

- `backup_configuration` - (Optional) Switch to enable/disable controller CloudN backup config. Valid values: true, false. Default value: false.
- `backup_cloud_type` - (Optional) Type of cloud service provider, requires an integer value. Use 1 for AWS.
- `backup_account_name` - (Optional) Name of the cloud account in the Aviatrix controller.
- `backup_bucket_name` - (Optional) S3 Bucket Name for AWS.
- `multiple_backups` - (Optional) Switch to enable the Controller to backup up to a maximum of 3 rotating backups. Valid values: true, false. Default value: false.

» Misc.

- `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 see the Controller upgrade guide 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
```