» opennebula_group

Use this data source to retrieve the group information for a given name.

» Example Usage

```
data "opennebula_group" "ExistingGroup" {
  name = "My_Service_Group"
}
```

» Argument Reference

- `name` - (Required) The OpenNebula group to retrieve information for.

» opennebula_image

Use this data source to retrieve the image information for a given name.

» Example Usage

```
data "opennebula_image" "ExistingImagr" {
  name = "My_Image"
}
```

» Argument Reference

- `name` - (Required) The OpenNebula image to retrieve information for.

» opennebula_security_group

Use this data source to retrieve the security group information for a given name.

» Example Usage

```
data "opennebula_security_group" "ExistingSecurityGroup" {
  name = "My_Security_Group"
}
```

» Argument Reference

- `name` - (Required) The OpenNebula security group to retrieve information for.

» opennebula_template

Use this data source to retrieve the template information for a given name.

» Example Usage

```
data "opennebula_template" "ExistingTemplate" {
  name = "My_Template"
}
```

» Argument Reference

- `name` - (Required) The OpenNebula template to retrieve information for.

» opennebula_virtual_data_center

Use this data source to retrieve the virtual data center information for a given name.

» Example Usage

```
data "opennebula_virtual_data_center" "ExistingVdc" {
  name = "My_VDC"
}
```

» Argument Reference

- `name` - (Required) The OpenNebula virtual data center to retrieve information for.

» opennebula_virtual_network

Use this data source to retrieve the virtual network information for a given name.

» Example Usage

```
data "opennebula_virtual_network" "ExistingVNet" {
  name = "My_VNet"
}
```

» Argument Reference

- `name` - (Required) The OpenNebula virtual network to retrieve information for.

» opennebula_group

Provides an OpenNebula group resource.

This resource allows you to manage groups on your OpenNebula clusters. When applied, a new group will be created. When destroyed, that group will be removed.

» Example Usage

```
data "template_file" "grptpl" {
  template = "${file("group_template.txt")}"
}

resource "opennebula_group" "group" {
    name = "terraform"
    template = "${data.template_file.grptpl.rendered}"
    delete_on_destruction = true
    quotas {
        datastore_quotas {
            id = 1
            images = 3
            size = 10000
        }
        vm_quotas {
            cpu = 3
```

```
            running_cpu = 3
            memory = 2048
            running_memory = 2048
        }
        network_quotas = {
            id = 10
            leases = 6
        }
        network_quotas = {
            id = 11
            leases = 4
        }
        image_quotas = {
            id = 8
            running_vms = 1
        }
        image_quotas = {
            id = 9
            running_vms = 1
        }
    }
}
```

with `group_template.txt` file with Sunstone information: `php SUNSTONE = [ DEFAULT_VIEW = "cloud",  GROUP_ADMIN_DEFAULT_VIEW = "groupadmin", GROUP_ADMIN_VIEWS = "cloud,groupadmin",  VIEWS = "cloud" ]`

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the group.
- `template` - (Required) Group template content in OpenNebula XML or String format. Used to provide SUSNTONE arguments.
- `delete_on_destruction` - (Optional) Flag to delete the group on destruction. Defaults to `false`.
- `admins` - (Optional) List of Administrator user IDs part of the group.
- `quotas` - (Optional) See Quotas parameters below for details

» Quotas parameters

`quotas` supports the following arguments:

- `datastore_quotas` - (Optional) List of datastore quotas. See Datastore quotas parameters below for details.

- `network_quotas` - (Optional) List of network quotas. See Network quotas parameters below for details.
- `image_quotas` - (Optional) List of image quotas. See Image quotas parameters below for details
- `vm_quotas` - (Optional) See Virtual Machine quotas parameters below for details

» Datastore quotas parameters

`datastore` supports the following arguments:

- `id` - (Required) Datastore ID.
- `images` - (Optional) Maximum number of images allowed on the datastore. Defaults to `default quota`
- `size` - (Optional) Total size in MB allowed on the datastore. Defaults to `default quota`

» Network quotas parameters

`network` supports the following arguments:

- `id` - (Required) Network ID.
- `leases` - (Optional) Maximum number of ip leases allowed on the network. Defaults to `default quota`

» Image quotas parameters

`image` supports the following arguments:

- `id` - (Required) Image ID.
- `running_vms` - (Optional) Maximum number of Virtual Machines in `RUNNING` state with this image ID attached. Defaults to `default quota`

» Virtual Machine quotas parameters

`vm` supports the following arguments:

- `cpu` - (Optional) Maximum number of CPU allowed (in total). Defaults to `default quota`.
- `memory` - (Optional) Maximum memory (in MB) allowed (in total). Defaults to `default quota`.
- `vms` - (Optional) Maximum number of Virtual Machines allowed. Defaults to `default quota`.
- `running_cpu` - (Optional) Number of CPUs of Virtual Machine in `RUNNING` state allowed. Defaults to `default quota`.
- `running_memory` - (Optional) Memory (in MB) of Virtual Machine in `RUNNING` state allowed. Defaults to `default quota`.

- `running_vms` - (Optional) Number of Virtual Machines in `RUNNING` state allowed. Defaults to `default quota`.
- `system_disk_size` - (Optional) Maximum disk size (in MB) on a `SYSTEM` datastore allowed (in total). Defaults to `default quota`.

» Attribute Reference

The following attribute is exported: * `id` - ID of the image.

» Import

To import an existing group #134 into Terraform, add this declaration to your .tf file:

```
resource "opennebula_group" "importgroup" {
    name = "importedgroup"
}
```

And then run:

```
terraform import opennebula_group.importgroup 134
```

Verify that Terraform does not perform any change:

```
terraform plan
```

» opennebula_image

Provides an OpenNebula image resource.

This resource allows you to manage images on your OpenNebula clusters. When applied, a new image will be created. When destroyed, that image will be removed.

» Example Usage

Clone an existing image and make it persistent: `hcl resource "opennebula_image" "osimageclone" {  clone_from_image = 12937  name = "terraclone-image" datastore_id = 113  persistent = true  permissions = "660" group = "terraform" }`

Allocate a new OS image using a URL: `hcl resource "opennebula_image" "osimage" {  name = "Ubuntu 18.04"  description = "Terraform image"  datastore_id = 103  persistent = false  lock = "MANAGE"  path = "http://marketplace.opennebula.org/appliance/ca5c3632-359a-429c-ac5b-b86`

```
dev_prefix = "vd"     driver = "qcow2"     permissions = "660"
group = "terraform" }
```

Allocate a new persistent 1GB datablock image: `hcl resource "opennebula_image" "datablockimage" {     name = "terra-datablock"    description = "Terraform datablock"    datastore_id = 103    persistent = true    type = "DATABLOCK"    size = "1000"    dev_prefix = "vd" driver = "qcow2"    group = "terraform" }`

Allocate a new context file: `hcl resource "opennebula_image" "contextfile" {     name = "terra-contextfile"     description = "Terraform context"     datastore_id = 2     type = "CONTEXT"     path = "http://server/myscript.sh" }`

Allocate a new CDROM image: `hcl resource "opennebula_image" "cdimage" {     name = "terra-cdimage"     description = "Terraform cdrom"     datastore_id = 103     type = "CDROM"     path = "http://server/mini.iso" }`

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the image.
- `description` - (Optional) Description of the image.
- `permissions` - (Optional) Permissions applied to the image. Defaults to the UMASK in OpenNebula (in UNIX Format: owner-group-other => Use-Manage-Admin.
- `clone_from_image` - (Optional) ID or name of the image to clone from. Conflicts with `path`, `size` and `type`.
- `datastore_id` - (Required) ID of the datastore to host new image. The `datastore_id` must be an `IMAGE` datastore.
- `persistent` - (Optional) Flag which indicates if the Image has to be persistent. Defaults to `false`.
- `lock` - (Optional) Lock the image with a specific lock level. Supported values: `USE`, `MANAGE`, `ADMIN`, `ALL` or `UNLOCK`.
- `path` - (Optional) Path or URL of the orinal image to use. Conflicts with `clone_from_image`.
- `type` - (Optional) Type of the image. Supported values: `OS`, `CDROM`, `DATABLOCK`, `KERNEL`, `RAMDISK` or `CONTEXT`. Conflicts with `clone_from_image`.
- `size` - (Optional) Size of the image in MB. Conflicts with `clone_from_image`.
- `dev_prefix` - (Optional) Device prefix on Virtual Machine. Usually one of these: `hd`, `sd` or `vd`.
- `target` - (Optional) Device target on Virtual Machine.
- `driver` - (Optional) OpenNebula Driver to use.
- `format` - (Optional) Image format. Example: `raw`, `qcow2`.

- `group` - (Optional) Name of the group which owns the image. Defaults to the caller primary group.

» Attribute Reference

The following attributes are exported: * `id` - ID of the image. * `uid` - User ID whom owns the image. * `gid` - Group ID which owns the image. * `uname` - User Name whom owns the image. * `gname` - Group Name which owns the image.

» Import

To import an existing image #14 into Terraform, add this declaration to your .tf file:

```
resource "opennebula_image" "importimage" {
    name = "importedimage"
}
```

And then run:

```
terraform import opennebula_image.importimage 14
```

Verify that Terraform does not perform any change:

```
terraform plan
```

» opennebula_security_group

Provides an OpenNebula security group resource.

This resource allows you to manage security groups on your OpenNebula clusters. When applied, a new security group will be created. When destroyed, that security group will be removed.

» Example Usage

```
resource "opennebula_security_group" "mysecgroup" {
    name = "terrasec"
    description = "Terraform security group"
    group = "terraform"
    rule {
        protocol = "ALL"
        rule_type = "OUTBOUND"
    }
```

```
    rule {
        protocol = "TCP"
        rule_type = "INBOUND"
        range = "22"
    }
    rule {
        protocol = "ICMP"
        rule_type = "INBOUND"
    }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the security group.
- `description` - (Optional) Description of the security group.
- `permissions` - (Optional) Permissions applied on security group. Defaults to the UMASK in OpenNebula (in UNIX Format: owner-group-other => Use-Manage-Admin.
- `commit` - (Optional) Flag to commit changes on Virtual Machine on security group update. Defaults to `true`.
- `rule` - (Required) List of rules. See Rules parameters below for details
- `group` - (Optional) Name of the group which owns the security group. Defaults to the caller primary group.

» Rules parameters

`rules` supports the following arguments:

- `protocol` - (Required) Protocol for the rule. Supported values: `ALL`, `TCP`, `UDP`, `ICMP` or `IPSEC`.
- `rule_type` - (Required) Direction of the traffic flow to allow, must be INBOUND or OUTBOUND.
- `network_id` - (Optional) VNET ID to be used as the source/destination IP addresses.
- `ip` - (Optional) IP (or starting IP if used with 'size') to apply the rule to.
- `size` - (Optional) Number of IPs to apply the rule from, starting with 'ip'.
- `range` - (Optional) Comma separated list of ports and port ranges.
- `icmp_type` - (Optional) Type of ICMP traffic to apply to when 'protocol' is ICMP.

See https://docs.opennebula.org/5.8/operation/network_management/security_groups.html for more details on allowed values.

» Attribute Reference

The following attribute are exported: * `id` - ID of the security group. * `uid` - User ID whom owns the security group. * `gid` - Group ID which owns the security group. * `uname` - User Name whom owns the security group. * `gname` - Group Name which owns the security group.

» Import

To import an existing security group #142 into Terraform, add this declaration to your .tf file:

```
resource "opennebula_security_group" "importsg" {
    name = "importedsg"
}
```

And then run:

```
terraform import opennebula_security_group.importsg 142
```

Verify that Terraform does not perform any change:

```
terraform plan
```

» opennebula_template

Provides an OpenNebula template resource.

This resource allows you to manage templates on your OpenNebula clusters. When applied, a new template will be created. When destroyed, that template will be removed.

» Example Usage

```
data "template_file" "templatetpl" {
  template = "${file("template-tpl.txt")}"
}

resource "opennebula_template" "mytemplate" {
  name = "mytemplate"
  template = "${data.template_file.templatetpl.rendered}"
  group = "terraform"
  permissions = "660"
}
```

with template file `template-tpl.txt`: `php CPU = 1 VCPU = 1 MEMORY = 512 Context = [ DNS_HOSTNAME = "YES", NETWORK = "YES", SSH_PUBLIC_KEY = "$USER[SSH_PUBLIC_KEY]" ] NIC_DEFAULT = [ MODEL = "virtio-net-pci" ] OS = [ ARCH = "x86_64", BOOT = "" ]`

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the virtual machine template.
- `template` - (Required) Text describing the OpenNebula template object, in Opennebula's XML string format.
- `permissions` - (Optional) Permissions applied on template. Defaults to the UMASK in OpenNebula (in UNIX Format: owner-group-other => Use-Manage-Admin.
- `group` - (Optional) Name of the group which owns the template. Defaults to the caller primary group.

» Attribute Reference

The following attribute are exported: * `id` - ID of the template. * `uid` - User ID whom owns the template. * `gid` - Group ID which owns the template. * `uname` - User Name whom owns the template. * `gname` - Group Name which owns the template. * `reg_time` - Registration time of the template.

» Import

To import an existing virtual machine template #54 into Terraform, add this declaration to your .tf file:

```
resource "opennebula_template" "importtpl" {
    name = "importedtpl"
}
```

And then run:

```
terraform import opennebula_template.importtppl 54
```

Verify that Terraform does not perform any change:

```
terraform plan
```

» opennebula_virtual_data_center

Provides an OpenNebula virtual data center resource.

This resource allows you to manage virtual data centers on your OpenNebula organization. When applied, a new virtual data center will be created. When destroyed, that virtual data center will be removed.

» Example Usage

```
resource "opennebula_virtual_data_center" "vdc" {
    name = "terravdc"
    group_ids = ["${opennebula_group.group.id}"]
    zones {
        id = 0
        host_ids = [0, 1]
        datastore_ids = [0, 1, 2]
        vnet_ids = ["${opennebula_virtual_network.vnet.id}"]
        cluster_ids = [0, 100]
    }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the virtual data center.
- `groups_ids` - (Optional) List of group IDs part of the virtual data center.
- `zones` - (Optional) List of zones. See Zones parameters below for details

» Zones parameters

`zones` supports the following arguments:

- `id` - (Optional) Zone ID from where resource to add in virtual data center are located. Defaults to 0.
- `host_ids` - (Optional) List of hosts from Zone ID to add in virtual data center.
- `datastore_ids` - (Optional) List of datastore from Zone ID to add in virtual data center.
- `vnet_ids` - (Optional) List of virtual networks from Zone ID to add in virtual data center.
- `cluster_ids` - (Optional) List of clusters from Zone ID to add in virtual data center.

» Attribute Reference

The following attribute is exported: * `id` - ID of the virtual data center.

» Import

To import an existing virtual data center #13 into Terraform, add this declaration to your .tf file:

```
resource "opennebula_virtual_data_center" "importvdc" {
    name = "importedvdc"
}
```

And then run:

```
terraform import opennebula_virtual_data_center.importvdc 13
```

Verify that Terraform does not perform any change:

```
terraform plan
```

» opennebula_virtual_machine

Provides an OpenNebula virtual machine resource.

This resource allows you to manage virtual machines on your OpenNebula clusters. When applied, a new virtual machine will be created. When destroyed, that virtual machine will be removed.

» Example Usage

```
data "template_file" "cloudinit" {
  template = "${file("cloud-init.yaml")}"
}

resource "opennebula_virtual_machine" "demo" {
  count = 2
  name = "tfdemovm"
  cpu = 1
  vcpu = 1
  memory = 1024
  group = "terraform"
  permissions = "660"

  context {
```

```
    NETWORK = "YES"
    HOSTNAME = "$NAME"
    USER_DATA = "${data.template_file.cloudinit.rendered}"
  }

  graphics {
    type = "VNC"
    listen = "0.0.0.0"
    keymap = "fr"
  }

  os {
    arch = "x86_64"
    boot = "disk0"
  }

  disk {
    image_id = "${opennebula_image.osimage.id}"
    size = 10000
    target = "vda"
    driver = "qcow2"
  }

  nic {
    model = "virtio-pci-net"
    network_id = "${var.vnetid}"
    security_groups = ["${opennebula_security_group.mysecgroup.id}"]
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the virtual machine.
- `permissions` - (Optional) Permissions applied on virtual machine. Defaults to the UMASK in OpenNebula (in UNIX Format: owner-group-other => Use-Manage-Admin.
- `template_id` - (Optional) If set, VM are instantiated from the template ID.
- `pending` - (Optional) Pending state during VM creation. Defaults to `false`.
- `cpu` - (Optional) Amount of CPU shares assigned to the VM. **Mandatory if ‘template_id‘ is not set**.
- `vpcu` - (Optional) Number of CPU cores presented to the VM.

- `memory` - (Optional) Amount of RAM assigned to the VM in MB. **Mandatory if 'template_id' is not set**.
- `context` - (Optional) Array of free form key=value pairs, rendered and added to the CONTEXT variables for the VM. Recommended to include at a minimum: NETWORK = "YES" and SET_HOSTNAME = "$NAME.
- `graphics` - (Optional) See Graphics parameters below for details.
- `os` - (Optional) See OS parameters below for details.
- `disk` - (Optional) Can be specified multiple times to attach several disks. See Disks parameters below for details.
- `nic` - (Optional) Can be specified multiple times to attach several NICs. See Nic parameters below for details.
- `group` - (Optional) Name of the group which owns the virtual machine. Defaults to the caller primary group.

» Graphics parameters

`graphics` supports the following arguments:

- `type` - (Required) Generally set to VNC.
- `listen` - (Required) Binding address.
- `port` - (Optional) Binding Port.
- `keymap` - (Optional) Keyboard mapping.

» OS parameters

`os` supports the following arguments:

- `arch` - (Required) Hardware architecture of the Virtual machine. Must fit the host architecture.
- `boot` - (Optional) `OS` disk to use to boot on.

» Disk parameters

`disk` supports the following arguments

- `image_id` - (Required) ID of the image to attach to the virtual machine.
- `size` - (Optional) Size (in MB) of the image attached to the virtual machine. Not possible to change a cloned image size.
- `target` - (Optional) Target name device on the virtual machine. Depends of the image `dev_prefix`.
- `driver` - (Optional) OpenNebula image driver.

Minimum 1 item. Maximum 8 items.

» NIC parameters

`nic` supports the following arguments

- `network_id` - (Required) ID of the virtual network to attach to the virtual machine.
- `ip` - (Optional) IP of the virtual machine on this network.
- `mac` - (Optional) MAC of the virtual machine on this network.
- `model` - (Optional) Nic model driver. Example: `virtio`.
- `physical_device` - (Optional) Physical device hosting the virtual network.
- `security_groups` - (Optional) List of security group IDs to use on the virtual network.

Minimum 1 item. Maximum 8 items.

» Attribute Reference

The following attribute are exported: * `id` - ID of the virtual machine. * `instance` - (Deprecated) Name of the virtual machine instance created on the cluster. * `uid` - User ID whom owns the virtual machine. * `gid` - Group ID which owns the virtual machine. * `uname` - User Name whom owns the virtual machine. * `gname` - Group Name which owns the virtual machine. * `state` - State of the virtual machine. * `lcmstate` - LCM State of the virtual machine.

» Import

To import an existing virtual machine #42 into Terraform, add this declaration to your .tf file:

```
resource "opennebula_virtual_machine" "importvm" {
    name = "importedvm"
}
```

`And then run:`

terraform import opennebula_virtual_machine.importvm 42 “‘

Verify that Terraform does not perform any change:

```
terraform plan
```

» opennebula_virtual_network

Provides an OpenNebula virtual network resource.

This resource allows you to manage virtual networks on your OpenNebula clusters. When applied, a new virtual network will be created. When destroyed, that virtual network will be removed.

» Example Usage

» Reservation of a virtual network

Allocate a new virtual network from the parent virtual network ”394”:

```
resource "opennebula_virtual_network" "reservation" {
    name = "terravnetres"
    description = "my terraform vnet"
    reservation_vnet = 394
    reservation_size = 5
    security_groups = [ 0 ]
}
```

» Virtual network creation

```
resource "opennebula_virtual_network" "vnet" {
    name = "tarravnet"
    permissions = "660"
    group = "${opennebula_group.group.name}"
    bridge = "br0"
    physical_device = "eth0"
    type = "fw"
    mtu = 1500
    ar = [ {
         ar_type = "IP4",
         size = 16
         ip4 = "172.16.100.101"
    } ]
    dns = "172.16.100.1"
    gateway = "172.16.100.1"
    security_groups = [ 0 ]
    clusters = [{
        id = 0
    }]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the virtual network.
- `description` - (Optional) Description of the virtual network.
- `permissions` - (Optional) Permissions applied on virtual network. Defaults to the UMASK in OpenNebula (in UNIX Format: owner-group-other => Use-Manage-Admin.
- `reservation_vnet` - (Optional) ID of the parent virtual network to reserve from. Conflicts with all parameters excepted `name`, `description`, `permissions`, `security_groups` and `group`.
- `reservation_size` - (Optional) Size (in address) reserved. Conflicts with all parameters excepted `name`, `description`, `permissions`, `security_groups` and `group`.
- `security_groups` - (Optional) List of security group IDs to apply on the virtual network.
- `bridge` - (Optional) Name of the bridge interface to which the virtual network should be associated. Conflicts with `reservation_vnet` and `reservation_size`.
- `physical_device` - (Optional) Name of the physical device interface to which the virtual network should be associated. Conflicts with `reservation_vnet` and `reservation_size`.
- `type` - (Optional) Virtual network type. One of these: `dummy`, `bridge`'`fw`, `ebtables`, `802.1Q`, `vxlan` or `ovswitch`. Defaults to `bridge`. Conflicts with `reservation_vnet` and `reservation_size`.
- `clusters` - (Optional) List of cluster IDs where the virtual network can be use. Conflicts with `reservation_vnet` and `reservation_size`.
- `vlan_id` - (Optional) ID of VLAN. Only if `type` is `802.1Q`, `vxlan` or `ovswitch`. Conflicts with `reservation_vnet`, `reservation_size` and `automatic_vlan_id`.
- `automatic_vlan_id` - (Optional) Flag to let OpenNebula scheduler to attribute the VLAN ID. Conflicts with `reservation_vnet`, `reservation_size` and `vlan_id`.
- `mtu` - (Optional) Virtual network MTU. Defaults to `1500`. Conflicts with `reservation_vnet` and `reservation_size`.
- `guest_mtu` - (Optional) MTU of the network caord on the virtual machine. **Cannot be greater than `mtu`**. Defaults to `1500`. Conflicts with `reservation_vnet` and `reservation_size`.
- `gateway` - (Optional) IP of the gateway. Conflicts with `reservation_vnet` and `reservation_size`.
- `network_mask` - (Optional) Network mask. Conflicts with `reservation_vnet` and `reservation_size`.
- `dns` - (Optional) Text String containing a comma separated list of DNS IPs. Conflicts with `reservation_vnet` and `reservation_size`.
- `ar` - (Optional) List of address ranges. See Address Range Parameters below for more details. Conflicts with `reservation_vnet` and `reservation_size`.
- `hold_size` - (Optional) Carve a network reservation of this size from the reservation starting from `ip_hold`. Conflicts with `reservation_vnet` and

`reservation_size`.

- `ip_hold` - (Optional) Start IP of the range to be held. Conflicts with `reservation_vnet` and `reservation_size`.
- `group` - (Optional) Name of the group which owns the virtual network. Defaults to the caller primary group.

» Address Range parameters

`ar` supports the following arguments:

- `ar_type` - (Optional) Address range type. Supported values: `IP4`, `IP6`, `IP6_STATIC`, `IP4_6` or `IP4_6_STATIC` or `ETHER`. Defaults to `IP4`.
- `ip4` - (Optional) Starting IPv4 address of the range. Required if `ar_type` is `IP4` or `IP4_6`.
- `ip6` - (Optional) Starting IPv6 address of the range. Required if `ar_type` is `IP6_STATIC` or `IP4_6_STATIC`.
- ‘size - (Optional) Address range size.
- `mac` - (Optional) Starting MAC Address of the range.
- `global_prefix` - (Optional) Global prefix for `IP6` or `IP_4_6`.
- `ula_prefix` - (Optional) ULA prefix for `IP6` or `IP_4_6`.
- `prefix_length` - (Optional) Prefix length. Only needed for `IP6_STATIC` or `IP4_6_STATIC`

» Attribute Reference

The following attribute are exported: * `id` - ID of the virtual network. * `uid` - User ID whom owns the virtual network. * `gid` - Group ID which owns the virtual network. * `uname` - User Name whom owns the virtual network. * `gname` - Group Name which owns the virtual network.

» Import

To import an existing virtual network #1234 into Terraform, add this declaration to your .tf file (don’t specify the reservation_size):

```
resource "opennebula_virtual_network" "importtest" {
    name = "importedvnet"
    reservation_vnet = 394
    # Security group "0" allows open access
    security_groups = ["0"]
}
```

And then run:

```
terraform import opennebula_virtual_network.importtest 1234
```

Verify that Terraform does not perform any change:

```
terraform plan
```