Hetzner Cloud Provider

The Hetzner Cloud (hcloud) provider is used to interact with the resources supported by Hetzner Cloud (https://www.hetzner.com/cloud). The provider needs to be configured with the proper credentials before it can be used.

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

Example Usage

```
# Set the variable value in *.tfvars file
# or using the -var="hcloud_token=..." CLI option
variable "hcloud_token" {}

# Configure the Hetzner Cloud Provider
provider "hcloud" {
  token = "${var.hcloud_token}"
}

# Create a server
resource "hcloud_server" "web" {
  # ...
}
```

Argument Reference

The following arguments are supported:

- `token` - (Required, string) This is the Hetzner Cloud API Token, can also be specified with the `HCLOUD_TOKEN` environment variable.
- `endpoint` - (Optional, string) Hetzner Cloud API endpoint, can be used to override the default API Endpoint `https://api.hetzner.cloud/v1`.
- `poll_interval` - (Optional, string) Configures the interval in which actions are polled by the client. Default `500ms`. Increase this interval if you run into rate limiting errors.

Data Source: hcloud_datacenter

Provides details about a specific Hetzner Cloud Datacenter. Use this resource to get detailed information about specific datacenter.

Example Usage

```
data "hcloud_datacenter" "ds_1" {
  name = "fsn1-dc8"
}
data "hcloud_datacenter" "ds_2" {
  id = 4
}
```

Argument Reference

- `id` - (Optional, string) ID of the datacenter.
- `name` - (Optional, string) Name of the datacenter.

Attributes Reference

- `id` - (int) Unique ID of the datacenter.
- `name` - (string) Name of the datacenter.
- `description` - (string) Description of the datacenter.
- `location` - (map) Physical datacenter location.
- `supported_server_type_ids` - (list) List of server types supported by the datacenter.
- `available_server_type_ids` - (list) List of available server types.

Data Source: hcloud_datacenters

Provides a list of available Hetzner Cloud Datacenters. This resource may be useful to create highly available infrastructure, distributed across several datacenters.

Example Usage

```
data "hcloud_datacenters" "ds" {
}

resource "hcloud_server" "workers" {
  name = "node1"
  image = "debian-9"
  server_type = "cx31"
  datacenter = "${element(data.hcloud_datacenters.ds.names, count.index)}"

  count = 3
}
```

Attributes Reference

- `datacenter_ids` - (list) List of unique datacenter identifiers.
- `names` - (list) List of datacenter names.
- `descriptions` - (list) List of all datacenter descriptions.

Data Source: hcloud_floating_ip

Provides details about a Hetzner Cloud Floating IP.

This resource can be useful when you need to determine a Floating IP ID based on the IP address.

Example Usage

Data Source: hcloud_floating_ip

Provides details about a Hetzner Cloud Floating IP. This resource can be useful when you need to determine a Floating IP ID based on the IP address.

Example Usage

```
data "hcloud_floating_ip" "ip_1" {
  ip_address = "1.2.3.4"
}
data "hcloud_floating_ip" "image_2" {
  with_selector = "key=value"
}
resource "hcloud_floating_ip_assignment" "main" {
  count          = "${var.counter}"
  floating_ip_id = "${data.hcloud_floating_ip.ip_1.id}"
  server_id      = "${hcloud_server.main.id}"
}
```

Argument Reference

- `ip_address` - (Optional, string) IP address of the Floating IP.
- `with_selector` - (Optional, string) Label selector (https://docs.hetzner.cloud/#overview-label-selector)

Attributes Reference

- `id` - (int) Unique ID of the Floating IP.
- `ip_address` - (string) IP address of the Floating IP.

Data Source: hcloud_image

Provides details about a Hetzner Cloud Image. This resource is useful if you want to use a non-terraform managed image.

Example Usage

```
data "hcloud_image" "image_1" {
  id = "1234"
}
data "hcloud_image" "image_2" {
  name = "ubuntu-18.04"
}
data "hcloud_image" "image_3" {
  with_selector = "key=value"
}

resource "hcloud_server" "main" {
  image  = "${data.hcloud_image.image_1.name}"
}
```

Argument Reference

- `id` - (Optional, string) ID of the Image.
- `name` - (Optional, string) Name of the Image.
- `with_selector` - (Optional, string) Label selector (https://docs.hetzner.cloud/#overview-label-selector)
- `most_recent` - (Optional, bool) If more than one result is returned, use the most recent Image.
- `with_status` - (Optional, list) List only images with the specified status, could contain `creating` or `available`.

Attributes Reference

- `id` - (int) Unique ID of the Image.
- `name` - (string) Name of the Image.
- `type` - (string) Type of the Image, could be `system`, `backup` or `snapshot`.
- `status` - (string) Status of the Image.
- `description` - (string) Description of the Image.
- `created` - (string) Date when the Image was created (in ISO-8601 format).
- `os_flavor` - (string) Flavor of operating system contained in the image, could be `ubuntu`, `centos`, `debian`, `fedora` or `unknown`.
- `os_version` - (string) Operating system version.

- `rapid_deploy` - (bool) Indicates that rapid deploy of the image is available.
- `deprecated` - (string) Point in time when the image is considered to be deprecated (in ISO-8601 format).

Data Source: hcloud_location

Provides details about a specific Hetzner Cloud Location. Use this resource to get detailed information about specific location.

Example Usage

```
data "hcloud_location" "l_1" {
  name = "fsn1"
}
data "hcloud_location" "l_2" {
  id = 1
}
```

Argument Reference

- `id` - (Optional, string) ID of the location.
- `name` - (Optional, string) Name of the location.

Attributes Reference

- `id` - (int) Unique ID of the location.
- `name` - (string) Name of the location.
- `description` - (string) Description of the location.
- `city` - (string) City of the location.
- `country` - (string) Country of the location.
- `latitude` - (float) Latitude of the city.
- `longitude` - (float) Longitude of the city.

Data Source: hcloud_locations

Provides a list of available Hetzner Cloud Locations. This resource may be useful to create highly available infrastructure, distributed across several locations.

Example Usage

```
data "hcloud_locations" "ds" {
}

resource "hcloud_server" "workers" {
  name = "node1"
  image = "debian-9"
  server_type = "cx31"
  location = "${element(data.hcloud_locations.ds.names, count.index)}"

  count = 3
}
```

Attributes Reference

- `location_ids` - (list) List of unique location identifiers.
- `names` - (list) List of location names.
- `descriptions` - (list) List of all location descriptions.

Data Source: hcloud_network

Provides details about a Hetzner Cloud network. This resource is useful if you want to use a non-terraform managed network.

Example Usage

```
data "hcloud_network" "network_1" {
  id = "1234"
}
data "hcloud_network" "network_2" {
  name = "my-network"
}
data "hcloud_network" "network_3" {
  with_selector = "key=value"
}
```

Argument Reference

- `id` - ID of the Network.
- `name` - Name of the Network.
- `with_selector` - Label Selector. For more information about possible values, visit the Hetzner Cloud Documentation (https://docs.hetzner.cloud/#overview-label-selector).

Attributes Reference

- `id` - Unique ID of the Network.
- `name` - Name of the Network.
- `ip_range` - IPv4 prefix of the Network.

Data Source: hcloud_server

Provides details about a Hetzner Cloud Server. This resource is useful if you want to use a non-terraform managed server.

Example Usage

```
data "hcloud_server" "s_1" {
  name = "my-server"
}
data "hcloud_server" "s_2" {
  id = "123"
}
data "hcloud_server" "s_3" {
  with_selector =  "key=value"
}
```

Argument Reference

- `id` - ID of the server.
- `name` - Name of the server.
- `with_selector` - Label Selector. For more information about possible values, visit the Hetzner Cloud Documentation (https://docs.hetzner.cloud/#overview-label-selector).
- `with_status` - (Optional, list) List only servers with the specified status, could contain `initializing`, `starting`, `running`, `stopping`, `off`, `deleting`, `rebuilding`, `migrating`, `unknown`.

Attributes Reference

- `id` - (int) Unique ID of the server.
- `name` - (string) Name of the server.
- `server_type` - (string) Name of the server type.
- `image` - (string) Name or ID of the image the server was created from.
- `location` - (string) The location name.
- `datacenter` - (string) The datacenter name.
- `backup_window` - (string) The backup window of the server, if enabled.
- `backups` - (boolean) Whether backups are enabled.
- `iso` - (string) ID or Name of the mounted ISO image.
- `ipv4_address` - (string) The IPv4 address.

- `ipv6_address` - (string) The first IPv6 address of the assigned network.
- `ipv6_network` - (string) The IPv6 network.
- `status` - (string) The status of the server.
- `labels` - (map) User-defined labels (key-value pairs)

Data Source: hcloud_ssh_key

Provides details about a Hetzner Cloud SSH Key. This resource is useful if you want to use a non-terraform managed SSH Key.

Example Usage

```
data "hcloud_ssh_key" "ssh_key_1" {
  id = "1234"
}
data "hcloud_ssh_key" "ssh_key_2" {
  name = "my-ssh-key"
}
data "hcloud_ssh_key" "ssh_key_3" {
  fingerprint = "43:51:43:a1:b5:fc:8b:b7:0a:3a:a9:b1:0f:66:73:a8"
}
data "hcloud_ssh_key" "ssh_key_4" {
  with_selector = "key=value"
}
resource "hcloud_server" "main" {
  ssh_keys  = ["${data.hcloud_ssh_key.ssh_key_1.id}","${data.hcloud_ssh_key.ssh_key_2.id}","${data.hcloud
_ssh_key.ssh_key_3.id}"]
}
```

Argument Reference

- `id` - (Optional, string) ID of the SSH Key.
- `name` - (Optional, string) Name of the SSH Key.
- `fingerprint` - (Optional, string) Fingerprint of the SSH Key.
- `with_selector` - (Optional, string) Label selector (https://docs.hetzner.cloud/#overview-label-selector)

Attributes Reference

- `id` - (int) Unique ID of the SSH Key.
- `name` - (string) Name of the SSH Key.
- `fingerprint` - (string) Fingerprint of the SSH Key.
- `public_key` - (string) Public Key of the SSH Key.

Data Source: hcloud_ssh_keys

Provides details about Hetzner Cloud SSH Keys. This resource is useful if you want to use a non-terraform managed SSH Key.

Example Usage

```
data "hcloud_ssh_keys" "all_keys" {
}
data "hcloud_ssh_keys" "keys_by_selector" {
  with_selector = "foo=bar"
}
resource "hcloud_server" "main" {
  ssh_keys  = "${data.hcloud_ssh_keys.all_keys.ssh_keys.*.name}"
}
```

Argument Reference

- `with_selector` - (Optional, string) Label selector (https://docs.hetzner.cloud/#overview-label-selector)

Attributes Reference

- `ssh_keys` - (list) List of all matches SSH keys. See `data.hcloud_ssh_key` for schema.

Data Source: hcloud_volume

Provides details about a Hetzner Cloud volume. This resource is useful if you want to use a non-terraform managed volume.

Example Usage

```
data "hcloud_volume" "volume_1" {
  id = "1234"
}
data "hcloud_volume" "volume_2" {
  name = "my-volume"
}
data "hcloud_volume" "volume_3" {
  with_selector = "key=value"
}
```

Argument Reference

- `id` - ID of the volume.
- `name` - Name of the volume.
- `with_selector` - Label Selector. For more information about possible values, visit the Hetzner Cloud Documentation (https://docs.hetzner.cloud/#overview-label-selector).
- `with_status` - (Optional, list) List only volumes with the specified status, could contain `creating` or `available`.

Attributes Reference

- `id` - Unique ID of the volume.
- `name` - Name of the volume.
- `size` - Size of the volume.

hcloud_floating_ip_assignment

Provides a Hetzner Cloud Floating IP Assignment to assign a Floating IP to a Hetzner Cloud Server. Deleting a Floating IP Assignment will unassign the Floating IP from the Server.

Example Usage

```
resource "hcloud_floating_ip_assignment" "main" {
  floating_ip_id = "${hcloud_floating_ip.master.id}"
  server_id = "${hcloud_server.node1.id}"
}

resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
  datacenter = "fsn1-dc8"
}

resource "hcloud_floating_ip" "master" {
  type = "ipv4"
  home_location = "nbg1"
}
```

Argument Reference

- `floating_ip_id` - (Required, int) ID of the Floating IP.
- `server_id` - (Required, int) Server to assign the Floating IP to.

Attributes Reference

- `id` - (int) Unique ID of the Floating IP Assignment.
- `floating_ip_id` - (int) ID of the Floating IP.
- `server_id` - (int) Server the Floating IP was assigned to.

hcloud_floating_ip

Provides a Hetzner Cloud Floating IP to represent a publicly-accessible static IP address that can be mapped to one of your servers.

Example Usage

```
resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
}

resource "hcloud_floating_ip" "master" {
  type = "ipv4"
  server_id = "${hcloud_server.node1.id}"
}
```

Argument Reference

- `type` - (Required, string) Type of the Floating IP. `ipv4` `ipv6`
- `server_id` - (Optional, int) Server to assign the Floating IP to.
- `home_location` - (Optional, string) Home location (routing is optimized for that location). Optional if server_id argument is passed.
- `description` - (Optional, string) Description of the Floating IP.
- `labels` - (Optional, map) User-defined labels (key-value pairs) should be created with.

Attributes Reference

- `id` - (int) Unique ID of the Floating IP.
- `type` - (string) Type of the Floating IP.
- `server_id` - (int) Server to assign the Floating IP is assigned to.
- `home_location` - (string) Home location.
- `description` - (string) Description of the Floating IP.
- `ip_address` - (string) IP Address of the Floating IP.
- `ip_network` - (string) IPv6 subnet. (Only set if `type` is `ipv6`)
- `labels` - (map) User-defined labels (key-value pairs)

Import

Floating IPs can be imported using its `id` :

```
terraform import hcloud_floating_ip.myip <id>
```

hcloud_network

Provides a Hetzner Cloud Network to represent a Network in the Hetzner Cloud.

Example Usage

```
resource "hcloud_network" "privNet" {
  name = "my-net"
  ip_range = "10.0.1.0/24"
}
```

Argument Reference

- `name` - (Required, string) Name of the Network to create (must be unique per project).
- `ip_range` - (Required, string) IP Range of the whole Network which must span all included subnets and route destinations. Must be one of the private ipv4 ranges of RFC1918.
- `labels` - (Optional, map) User-defined labels (key-value pairs) should be created with.

Attributes Reference

- `id` - (int) Unique ID of the network.
- `name` - (string) Name of the network.
- `ip_range` - (string) IPv4 Prefix of the whole Network.
- `labels` - (map) User-defined labels (key-value pairs)

Import

Networks can be imported using its `id` :

```
terraform import hcloud_network.myip <id>
```

hcloud_network_route

Provides a Hetzner Cloud Network Route to represent a Network route in the Hetzner Cloud.

Example Usage

```
resource "hcloud_network" "mynet" {
  name = "my-net"
  ip_range = "10.0.0.0/8"
}
resource "hcloud_network_route" "privNet" {
  network_id = "${hcloud_network.mynet.id}"
  destination = "10.100.1.0/24"
  gateway = "10.0.1.1"
}
```

Argument Reference

- `network_id` - (Required, int) ID of the Network the route should be added to.
- `destination` - (Required, string) Destination network or host of this route. Must be a subnet of the ip_range of the Network. Must not overlap with an existing ip_range in any subnets or with any destinations in other routes or with the first ip of the networks ip_range or with 172.31.1.1.
- `gateway` - (Required, string) Gateway for the route. Cannot be the first ip of the networks ip_range and also cannot be 172.31.1.1 as this IP is being used as a gateway for the public network interface of servers.

Attributes Reference

- `id` - (int) Unique ID of the Network route.
- `network_id` - (int) ID of the Network.
- `destination` - (string) Destination of this route.
- `gateway` - (string) Gateway of the route.

hcloud_network_subnet

Provides a Hetzner Cloud Network Subnet to represent a Subnet in the Hetzner Cloud.

Example Usage

```
resource "hcloud_network" "mynet" {
  name = "my-net"
  ip_range = "10.0.0.0/8"
}
resource "hcloud_network_subnet" "foonet" {
  network_id = "${hcloud_network.mynet.id}"
  type = "server"
  network_zone = "eu-central"
  ip_range   = "10.0.1.0/24"
}
```

Argument Reference

- `network_id` - (Required, int) ID of the Network the subnet should be added to.
- `type` - (Required, string) Type of subnet. `server`
- `ip_range` - (Required, string) Range to allocate IPs from. Must be a subnet of the ip_range of the Network and must not overlap with any other subnets or with any destinations in routes.
- `network_zone` - (Required, string) Name of network zone.

Attributes Reference

- `id` - (string) ID of the Network subnet.
- `network_id` - (int) ID of the Network.
- `type` - (string) Type of subnet.
- `ip_range` - (string) Range to allocate IPs from.
- `network_zone` - (string) Name of network zone.

hcloud_rdns

Provides a Hetzner Cloud Reverse DNS Entry to create, modify and reset reverse dns entries for Hetzner Cloud Floating IPs or servers.

Example Usage

For servers:

```
resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
}

resource "hcloud_rdns" "master" {
  server_id = "${hcloud_server.node1.id}"
  ip_address = "${hcloud_server.node1.ipv4_address}"
  dns_ptr = "example.com"
}
```

For Floating IPs:

```
resource "hcloud_floating_ip" "floating1" {
  home_location = "nbg1"
  type = "ipv4"
}

resource "hcloud_rdns" "floating_master" {
  floating_ip_id = "${hcloud_floating_ip.floating1.id}"
  ip_address = "${hcloud_floating_ip.floating1.ip_address}"
  dns_ptr = "example.com"
}
```

Argument Reference

- `dns_ptr` - (Required, string) The DNS address the `ip_address` should resolve to.
- `ip_address` - (Required, string) The IP address that should point to `dns_ptr`.
- `server_id` - (Required, int) The server the `ip_address` belongs to.
- `floating_ip_id` - (Required, int) The Floating IP the `ip_address` belongs to.

Attributes Reference

- `id` - (int) Unique ID of the Reverse DNS Entry.

- `dns_ptr` - (string) DNS pointer for the IP address.
- `ip_address` - (string) IP address.
- `server_id` - (int) The server the IP address belongs to.
- `floating_ip_id` - (int) The Floating IP the IP address belongs to.

Import

Reverse DNS entries can be imported using a compound ID with the following format: `<prefix (s for server/ f for floating ip)>-<server or floating ip ID>-<IP address>`

```
# import reverse dns entry on server with id 123, ip 192.168.100.1
terraform import hcloud_rdns.myrdns s-123-192.168.100.1

# import reverse dns entry on floating ip with id 123, ip 2001:db8::1
terraform import hcloud_rdns.myrdns f-123-2001:db8::1
```

hcloud_server

Provides an Hetzner Cloud server resource. This can be used to create, modify, and delete servers. Servers also support provisioning (https://www.terraform.io/docs/provisioners/index.html).

Example Usage

```
# Create a new server running debian
resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) Name of the server to create (must be unique per project and a valid hostname as per RFC 1123).
- `server_type` - (Required, string) Name of the server type this server should be created with.
- `image` - (Required, string) Name or ID of the image the server is created from.
- `location` - (Optional, string) The location name to create the server in. `nbg1`, `fsn1` or `hel1`
- `datacenter` - (Optional, string) The datacenter name to create the server in.
- `user_data` - (Optional, string) Cloud-Init user data to use during server creation
- `ssh_keys` - (Optional, list) SSH key IDs or names which should be injected into the server at creation time
- `keep_disk` - (Optional, bool) If true, do not upgrade the disk. This allows downgrading the server type later.
- `iso` - (Optional, string) ID or Name of an ISO image to mount.
- `rescue` - (Optional, string) Enable and boot in to the specified rescue system. This enables simple installation of custom operating systems. `linux64` `linux32` or `freebsd64`
- `labels` - (Optional, map) User-defined labels (key-value pairs) should be created with.
- `backups` - (Optional, boolean) Enable or disable backups.

Attributes Reference

The following attributes are exported:

- `id` - (int) Unique ID of the server.

- `name` - (string) Name of the server.
- `server_type` - (string) Name of the server type.
- `image` - (string) Name or ID of the image the server was created from.
- `location` - (string) The location name.
- `datacenter` - (string) The datacenter name.
- `backup_window` - (string) The backup window of the server, if enabled.
- `backups` - (boolean) Whether backups are enabled.
- `iso` - (string) ID or Name of the mounted ISO image.
- `ipv4_address` - (string) The IPv4 address.
- `ipv6_address` - (string) The first IPv6 address of the assigned network.
- `ipv6_network` - (string) The IPv6 network.
- `status` - (string) The status of the server.
- `labels` - (map) User-defined labels (key-value pairs)

Import

Servers can be imported using the server `id`:

```
terraform import hcloud_server.myserver <id>
```

hcloud_server_network

Provides a Hetzner Cloud Server Network to represent a private network on a server in the Hetzner Cloud.

Example Usage

```
resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
}
resource "hcloud_network" "mynet" {
  name = "my-net"
  ip_range = "10.0.0.0/8"
}
resource "hcloud_network_subnet" "foonet" {
  network_id = "${hcloud_network.mynet.id}"
  type = "server"
  network_zone = "eu-central"
  ip_range   = "10.0.1.0/24"
}

resource "hcloud_server_network" "srvnetwork" {
  server_id = "${hcloud_server.node1.id}"
  network_id = "${hcloud_network.mynet.id}"
  ip = "10.0.1.5"
}
```

Argument Reference

- `network_id` - (Required, int) ID of the network which should be added to the server.
- `server_id` - (Required, int) ID of the server.
- `ip` - (Optional, string) IP to request to be assigned to this server. If you do not provide this then you will be auto assigned an IP address.
- `alias_ips` - (Required, list[string]) Additional IPs to be assigned to this server.

Attributes Reference

- `id` - (string) ID of the server network.
- `network_id` - (int) ID of the network.
- `server_id` - (int) ID of the server.
- `ip` - (string) IP assigned to this server.

- `alias_ips` - (list[string]) Additional IPs assigned to this server.

hcloud_ssh_key

Provides a Hetzner Cloud SSH key resource to manage SSH keys for server access.

Example Usage

```
# Create a new SSH key
resource "hcloud_ssh_key" "default" {
  name = "Terraform Example"
  public_key = "${file("~/.ssh/id_rsa.pub")}"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) Name of the SSH key.
- `public_key` - (Required, string) The public key. If this is a file, it can be read using the file interpolation function

Attributes Reference

The following attributes are exported:

- `id` - (int) The unique ID of the key.
- `name` - (string) The name of the SSH key
- `public_key` - (string) The text of the public key
- `fingerprint` - (string) The fingerprint of the SSH key
- `labels` - (map) User-defined labels (key-value pairs)

Import

SSH keys can be imported using the SSH key `id` :

```
terraform import hcloud_ssh_key.mykey <id>
```

hcloud_volume_attachment

Provides a Hetzner Cloud Volume attachment to attach a Volume to a Hetzner Cloud Server. Deleting a Volume Attachment will detach the Volume from the Server.

Example Usage

```
resource "hcloud_volume_attachment" "main" {
  volume_id = "${hcloud_volume.master.id}"
  server_id = "${hcloud_server.node1.id}"
  automount = true
}

resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
  datacenter = "nbg1-dc3"
}

resource "hcloud_volume" "master" {
  location = "nbg1"
  size     = 10
}
```

Argument Reference

- `volume_id` - (Required, int) ID of the Volume.
- `server_id` - (Required, int) Server to attach the Volume to.
- `automount` - (Optional, bool) Automount the volume upon attaching it.

Attributes Reference

- `id` - (int) Unique ID of the Volume Attachment.
- `volume_id` - (int) ID of the Volume.
- `server_id` - (int) Server the Volume was attached to.

hcloud_volume

Provides a Hetzner Cloud volume resource to manage volumes.

Example Usage

```
resource "hcloud_server" "node1" {
  name = "node1"
  image = "debian-9"
  server_type = "cx11"
}

resource "hcloud_volume" "master" {
  name = "volume1"
  size = 50
  server_id = "${hcloud_server.node1.id}"
  automount = true
}
```

Argument Reference

- `name` - (Required, string) Name of the volume to create (must be unique per project).
- `size` - (Required, int) Size of the volume (in GB).
- `server` - (Optional, int) Server to attach the Volume to, optional if location argument is passed.
- `location` - (Optional, string) Location of the volume to create, optional if server_id argument is passed.
- `automount` - (Optional, bool) Automount the volume upon attaching it (server_id must be provided).
- `format` - (Optional, string) Format volume after creation. `xfs` or `ext4`

Note: When you want to attach multiple volumes to a server, please use the `hcloud_volume_attachment` resource and the `location` argument instead of the `server_id` argument.

Attributes Reference

- `id` - Unique ID of the volume.
- `name` - Name of the volume.
- `size` - Size of the volume.
- `labels` - User-defined labels (key-value pairs).
- `linux_device` - Device path on the file system for the Volume.

Import

Volumes can be imported using their `id` :

```
terraform import hcloud_volume.myvolume <id>
```