Terraform provider serverscom

Sep 10, 2020

Getting started

CHAPTER 1

Setup guide

This guide will help you to set up Servers.com as a Terraform provider. Follow the steps described below to create the correct provider's configuration and run it in a proper way.

1) Download Terraform from the official page and install according to the instructions.
2) After a successful installation, create a directory with any name to store Terraform configuration files. We take "serverscom-terraform" as an example.
3) Create a configuration file "main.tf" (it can have another name). The file must be located in the "serverscom-terraform" directory and have the ".tf" extension.
4) Write the following configuration in the "main.tf":

```
provider "serverscom" {
  token = "<your API token>"
  endpoint = "https://api.servers.com/v1"
}
```

   To see a description of the attributes, go to Servers.com Provider section.

5) Download Servers.com provider binary from this repository. Choose an appropriate archive by its name according to your operating system. All the files are named this way: `terraform-provider-serverscom-<version X.Y.Z>-<Operating system>-<Architecture>`

   For example: `terraform-provider-serverscom-v0.1.1-windows-amd64.zip`

6) Extract the archive in the following directory (it depends on an operating system):

- for Windows: `%APPDATA%\terraform.d\plugins\windows_amd64`
- for Linux: `~/.terraform.d/plugins/linux_amd64`
- for MacOS: `~/.terraform.d/plugins/darwin_amd64`

7) Run `terraform init` command in the directory "serverscom-terraform" to download a Servers.com plugin, it will be saved in the current directory. Thus, Terraform will be able to interact with Servers.com.

8) Terraform is ready to work with the Servers.com infrastructure, just complete the configuration file with a description of a desired infrastructure.

Here are some basic commands that may be useful:

- `terraform plan` - see what actions Terraform will make to get a described configuration;
- `terraform apply` - start a creating of an infrastructure according to a described configuration.

CHAPTER 2

Servers.com Provider

The Servers.com Provider allows to interact with Servers.com services. The provider has to be set up properly before using, we recommend you to get acquainted with the Getting started instruction. To see available resources and its description, use the navigation.

2.1 Example Usage

```
# Configure the Servers.com Provider
provider "serverscom" {
  token = "<your API token>"
  endpoint = "https://api.servers.com/v1"
}

# Create a dedicated server
resource "serverscom_dedicated_server" "node_1" {
  ...
}
```

2.2 Argument Reference

- `token` (Required, string) - The token used to perform API-requests for Servers.com services, it can be issued in the Customer Portal.
- `endpoint` (Optional, string) - The Servers.com API endpoint. In most of cases the default one is used: https://api.servers.com/v1.

CHAPTER 3

Resources

List of resources

3.1 Available Resources

The following list contains a description of the resources:

3.1.1 Dedicated server

Provides a Servers.com dedicated server resource. This can be used to create, modify, and delete servers. Servers also support provisioning.

Example

Create a new dedicated server:

```
resource "serverscom_dedicated_server" "node_1" {
  hostname             = "node-1"
  location             = "SJC1"
  server_model         = "Dell R440 / 2xIntel Xeon Silver-4114 / 32 GB RAM / 1x480 GB␣
↪SSD"
  ram_size             = 32

  operating_system     = "Ubuntu 16.04-server x86_64"

  private_uplink       = "Private 10 Gbps with redundancy"
  public_uplink        = "Public 10 Gbps with redundancy"
  bandwidth            = "200000 GB"

  ssh_key_fingerprints = [
    "cf:1d:09:ab:cb:47:97:3f:50:9a:f0:34:14:78:fa:1b"
```

(continues on next page)

(continued from previous page)

```
  ]

  ipv6 = true

  slot {
    drive_model = "480 GB SSD SATA"
    position    = 0
  }

  layout {
    slot_positions = [0]

    partition {
      target = "/"
      size = 10240
      fill = false
      fs = "ext4"
    }

    partition {
      target = "/home"
      size = 1
      fill = true
      fs = "ext4"
    }

    partition {
      target = "swap"
      size = 4096
      fill = false
    }
  }
}
```

Argument Reference

The following arguments are supported:

- *hostname* - (Required, string) Name of the dedicated server (according to RFC 1123 specification).
- *location* - (Required, string) Location code of the dedicated server. For example: *AMS1*, *SJC1*, etc.
- *server_model* - (Required, string) Name of the dedicated server model.
- *ram_size* - (Optional, int) Size of the RAM (GB).
- *operating_system* - (Optional, string) The dedicated server operating system name.
- *private_uplink* - (Required, string) The dedicated server private uplink name.
- *public_uplink* - (Optional, string) The dedicated server public uplink name.
- *bandwidth* - (Optional, string) The dedicated server public bandwidth name.
- *ssh_key_fingerprints.0* - (Optional, string) SSH key fingerprint.
- *ipv6* - (Optional, bool) Is IPv6 enabled. Defaults to *false*.
- *slot* - (Optional, list) List of drive slots.

- *slot.0.position* - (Required, int) Slot position.
- *slot.0.drive_model_name* - (Optional, string) The name of drive model to place in the slot.
- *layout* - (Optional, list) List of layouts.
- *layout.0.slot_positions* - (Required, list) List of slots which should be used in the layout.
- *layout.0.raid* - (Optional, int) RAID level for the layout.
- *layout.0.partition* - (Required, list) List of partitions for the layout.
- *layout.0.partition.0.target* - (Required, string) Target/Mount point for the partition.
- *layout.0.partition.0.size* - (Required, int) Size of the partition (MB).
- *layout.0.partition.0.fill* - (Optional, bool) Autofill partition by all unused space. When set to *true*, the size will be ignored.
- *layout.0.partition.0.fs* - (Optional, string) Filesystem for the partition.

Attributes Reference

The following attributes are exported:

- *id* - (string) Unique identifier of the dedicated server.
- *hostname* - (string) Name of the dedicated server.
- *location* - (string) Location code of the dedicated server. For example: *AMS1*, *SJC1*, etc.
- *server_model* - (string) Name of the server model.
- *ram_size* - (int) Size of the RAM (GB).
- *operating_system* - (string) The dedicated server operating system name.
- *private_uplink* - (string) The dedicated server private uplink name.
- *public_uplink* - (string) The dedicated server public uplink name.
- *bandwidth* - (string) The dedicated server public bandwidth name.
- *status* - (string) Status of the dedicated server.
- *private_ipv4_address* - (string) The private IPv4 address.
- *public_ipv4_address* - (string) The public IPv4 address.
- *configuration* - (string) The current configuration name of the dedicated server.
- *slots* - (list) List of drive slots in the dedicated server.
- *slots.0.position* - (int) Slot position.
- *slots.0.drive_model_name* - (string) Name of drive model.

Import

Dedicated servers can be imported using the dedicated server *id*:

```
terraform import serverscom_dedicated_server.node_1 <id>
```

3.1.2 Cloud computing instance

Provides an Servers.com cloud computing instance resource. This can be used to create, modify, and delete cloud computing instances. Cloud computing instances also support provisioning.

Example

Create a new cloud computing instance

```
resource "serverscom_cloud_computing_instance" "instance_1" {
  name = "instance-1"
  region = "NL01"
  image = "Ubuntu 20.04-server (64 bit)"

  flavor = "SSD.50"

  gpn_enabled = true
  ipv6_enabled = true
  backup_copies = 5

  ssh_key_fingerprint = "cf:1d:09:ab:cb:47:97:3f:50:9a:f0:34:14:78:fa:1b"
}
```

Argument Reference

The following arguments are supported:

- *name* - (Required, string) Name of the cloud instance (according to RFC 1123 specification).
- *region* - (Required, string) Cloud computing region code.
- *image* - (Required, string) Name of the image.
- *flavor* - (Required, string) Name of the flavor.
- *gpn_enabled* - (Optional, bool) Is GPN network enabled. Defaults to *false*.
- *ipv6_enabled* - (Optional, bool) Is IPv6 enabled. Defaults to *false*.
- *backup_copies* - (Optional, int) Count of backup copies. Defaults to *0*.
- *ssh_key_fingerprint* - (Optional, string) SSH key fingerprint.

Attributes Reference

The following attributes are exported:

- *id* - (string) Unique identifier of the cloud computing instance.
- *name* - (string) Name of the cloud instance (according to RFC 1123 specification).
- *region* - (string) Cloud computing region code.
- *image* - (string) Name of the image.
- *flavor* - (string) Name of the flavor.
- *gpn_enabled* - (bool) Is GPN network enabled. Defaults to *false*.
- *ipv6_enabled* - (bool) Is IPv6 enabled. Defaults to *false*.

- *backup_copies* - (int) Count of backup copies. Defaults to *0*.
- *status* - (string) Status of the cloud computing instance.
- *private_ipv4_address* - (string) Private IPv4 address.
- *public_ipv4_address* - (string) Public IPv4 address.
- *public_ipv6_address* - (string) Public IPv6 address.
- *openstack_uuid* - (string) OpenStack unique identifier (UUID) of the cloud computing instance.

Import

Cloud computing instances can be imported using the cloud computing instance *id*:

```
terraform import serverscom_cloud_computing_instance.instance_1 <id>
```

3.1.3 SSH Key

Provides a Servers.com SSH key resource to manage SSH keys for dedicated server/cloud computing instance access.

Example

Create a new SSH key

```
resource "serverscom_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) 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) Unique identifier of the SSH key.
- *name* - (string) Name of the SSH key.
- *public_key* - (string) Public part of the SSH key.
- *fingerprint* - (string) Fingerprint of the SSH key.

Import

SSH keys can be imported using the SSH key *fingeprint*:

```
terraform import serverscom_ssh_key.default <fingerprint>
```

3.1.4 L2 Segment

Provides an Servers.com l2 segment resource. This can be used to create, modify, and delete L2 segments.

Example

Create a new L2 segment

```
resource "serverscom_l2_segment" "segment_1" {
  name = "l2-segment-1"
  type = "private"
  location_group = "SJC1"

  member {
    id = "QBeXDWey"
    mode = "native"
  }

  member {
    id = "4QbYEKbz"
    mode = "native"
  }
}
```

Argument Reference

The following arguments are supported:

- *name* - (Optional, string) Name of the L2 segment.
- *type* - (Required, string) Type of the L2 segment.
- *location_group* - (Required, string) Location group code.
- *member* - (Required, list) List of the L2 segment members.
- *member.0.id* - (Required, int) ID of the dedicated server.
- *member.0.mode* - (Required, string) Membership mode of the dedicated server.

Attributes Reference

The following attributes are exported:

- *id* - (string) Unique identifier of the L2 segment.
- *name* - (string) Name of the L2 segment.
- *type* - (string) Type of the L2 segment.
- *location_group* - (string) Location group code.

- *member* - (list) List of the L2 segment members.
- *member.0.id* - (int) ID of the dedicated server.
- *member.0.mode* - (string) Membership mode of the dedicated server.
- *member.0.status* - (string) Status of the membership.
- *member.0.vlan* - (int) VLAN number of the member.
- *member.0.created_at* - (string) Member created at.
- *member.0.updated_at* - (string) Member updated at.
- *status* - (string) Status of the L2 segment.
- *created_at* - (string) L2 segment created at.
- *updated_at* - (string) L2 segment updated at.

Import

L2 Segments can be imported using the l2 segment *id*:

```
terraform import serverscom_l2_segment.segment_1 <id>
```