» openstack_compute_flavor_v2

Use this data source to get the ID of an available OpenStack flavor.

» Example Usage

```
data "openstack_compute_flavor_v2" "small" {
  vcpus = 1
  ram   = 512
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Compute client. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the flavor.
- `min_ram` - (Optional) The minimum amount of RAM (in megabytes).
- `ram` - (Optional) The exact amount of RAM (in megabytes).
- `min_disk` - (Optional) The minimum amount of disk (in gigabytes).
- `disk` - (Optional) The exact amount of disk (in gigabytes).
- `vcpus` - (Optional) The amount of VCPUs.
- `swap` - (Optional) The amount of swap (in gigabytes).
- `rx_tx_factor` - (Optional) The `rx_tx_factor` of the flavor.

» Attributes Reference

`id` is set to the ID of the found flavor. In addition, the following attributes are exported:

- `is_public` - Whether the flavor is public or private.

» openstack_dns_zone_v2

Use this data source to get the ID of an available OpenStack DNS zone.

» Example Usage

```
data "openstack_dns_zone_v2" "zone_1" {
  name = "example.com"
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 DNS client. A DNS client is needed to retrieve zone ids. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the zone.
- `description` - (Optional) A description of the zone.
- `email` - (Optional) The email contact for the zone record.
- `status` - (Optional) The zone's status.
- `ttl` - (Optional) The time to live (TTL) of the zone.
- `type` - (Optional) The type of the zone. Can either be `PRIMARY` or `SECONDARY`.

» Attributes Reference

`id` is set to the ID of the found zone. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `email` - See Argument Reference above.
- `type` - See Argument Reference above.
- `ttl` - See Argument Reference above.
- `description` - See Argument Reference above.
- `status` - See Argument Reference above.
- `attributes` - Attributes of the DNS Service scheduler.
- `masters` - An array of master DNS servers. When `type` is `SECONDARY`.
- `created_at` - The time the zone was created.
- `updated_at` - The time the zone was last updated.
- `transferred_at` - The time the zone was transferred.
- `version` - The version of the zone.
- `serial` - The serial number of the zone.
- `pool_id` - The ID of the pool hosting the zone.
- `project_id` - The project ID that owns the zone.

» openstack_identity_project_v3

Use this data source to get the ID of an OpenStack project.

» Example Usage

```
data "openstack_identity_project_v3" "project_1" {
  name = "demo"
}
```

» Argument Reference

The following arguments are supported:

- `domain_id` - (Optional) The domain this project belongs to.
- `enabled` - (Optional) Whether the project is enabled or disabled. Valid values are `true` and `false`.
- `is_domain` - (Optional) Whether this project is a domain. Valid values are `true` and `false`.
- `name` - (Optional) The name of the project.
- `parent_id` - (Optional) The parent of this project.

» Attributes Reference

`id` is set to the ID of the found project. In addition, the following attributes are exported:

- `description` - The description of the project.
- `domain_id` - See Argument Reference above.
- `enabled` - See Argument Reference above.
- `is_domain` - See Argument Reference above.
- `name` - See Argument Reference above.
- `parent_id` - See Argument Reference above.
- `region` - The region the project is located in.

» openstack_identity_role_v3

Use this data source to get the ID of an OpenStack role.

» Example Usage

```
data "openstack_identity_role_v3" "admin" {
  name = "admin"
}
```

» Argument Reference

- `name` - The name of the role.
- `domain_id` - (Optional) The domain the role belongs to.
- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used.

» Attributes Reference

`id` is set to the ID of the found role. In addition, the following attributes are exported:

- `name` - See Argument Reference above.
- `domain_id` - See Argument Reference above.
- `region` - See Argument Reference above.

» openstack_identity_user_v3

Use this data source to get the ID of an OpenStack user.

» Example Usage

```
data "openstack_identity_user_v3" "user_1" {
  name = "user_1"
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) A description of the user.
- `default_project_id` - (Optional) The default project this user belongs to.

- `domain_id` - (Optional) The domain this user belongs to.
- `enabled` - (Optional) Whether the user is enabled or disabled. Valid values are `true` and `false`.
- `idp_id` - (Optional) The identity provider ID of the user.
- `name` - (Optional) The name of the user.
- `password_expires_at` - (Optional) Query for expired passwords. See the OpenStack API docs for more information on the query format.
- `protocol_id` - (Optional) The protocol ID of the user.
- `unique_id` - (Optional) The unique ID of the user.

» Attributes Reference

The following attributes are exported:

- `default_project_id` - See Argument Reference above.
- `domain_id` - See Argument Reference above.
- `enabled` - See Argument Reference above.
- `idp_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `password_expires_at` - See Argument Reference above.
- `protocol_id` - See Argument Reference above.
- `region` - The region the user is located in.
- `unique_id` - See Argument Reference above.

» openstack_images_image_v2

Use this data source to get the ID of an available OpenStack image.

» Example Usage

```
data "openstack_images_image_v2" "ubuntu" {
  name = "Ubuntu 16.04"
  most_recent = true

  properties {
    key = "value"
  }
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Glance client. A Glance client is needed to create an Image that can be used with a compute instance. If omitted, the `region` argument of the provider is used.
- `most_recent` - (Optional) If more than one result is returned, use the most recent image.
- `name` - (Optional) The name of the image.
- `owner` - (Optional) The owner (UUID) of the image.
- `properties` - (Optional) a map of key/value pairs to match an image with. All specified properties must be matched.
- `size_min` - (Optional) The minimum size (in bytes) of the image to return.
- `size_max` - (Optional) The maximum size (in bytes) of the image to return.
- `sort_direction` - (Optional) Order the results in either `asc` or `desc`.
- `sort_key` - (Optional) Sort images based on a certain key. Defaults to `name`.
- `tag` - (Optional) Search for images with a specific tag.
- `visibility` - (Optional) The visibility of the image. Must be one of "public", "private", "community", or "shared". Defaults to "private".
- `member_status` - (Optional) The status of the image. Must be one of "accepted", "pending", "rejected", or "all".

» Attributes Reference

`id` is set to the ID of the found image. In addition, the following attributes are exported:

- `checksum` - The checksum of the data associated with the image.
- `created_at` - The date the image was created.
- `container_format`: The format of the image's container.
- `disk_format`: The format of the image's disk.
- `file` - the trailing path after the glance endpoint that represent the location of the image or the path to retrieve it.
- `metadata` - The metadata associated with the image. Image metadata allow for meaningfully define the image properties and tags. See http://docs.openstack.org/developer/glance/metadefs-concepts.html.
- `min_disk_gb` - The minimum amount of disk space required to use the image.
- `min_ram_mb` - The minimum amount of ram required to use the image.

- `properties` - Freeform information about the image.
- `protected` - Whether or not the image is protected.
- `schema` - The path to the JSON-schema that represent the image or image
- `size_bytes` - The size of the image (in bytes).
- `tags` - See Argument Reference above.
- `update_at` - The date the image was last updated.

» openstack_networking_network_v2

Use this data source to get the ID of an available OpenStack network.

» Example Usage

```
data "openstack_networking_network_v2" "network" {
  name = "tf_test_network"
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve networks ids. If omitted, the `region` argument of the provider is used.
- `network_id` - (Optional) The ID of the network.
- `name` - (Optional) The name of the network.
- `status` - (Optional) The status of the network.
- `matching_subnet_cidr` - (Optional) The CIDR of a subnet within the network.
- `tenant_id` - (Optional) The owner of the network.
- `availability_zone_hints` - (Optional) The availability zone candidates for the network.

» Attributes Reference

`id` is set to the ID of the found network. In addition, the following attributes are exported:

- `admin_state_up` - (Optional) The administrative state of the network.
- `name` - See Argument Reference above.
- `region` - See Argument Reference above.

- `shared` - (Optional) Specifies whether the network resource can be accessed by any tenant or not.
- `availability_zone_hints` - (Optional) The availability zone candidates for the network.

» openstack_networking_secgroup_v2

Use this data source to get the ID of an available OpenStack security group.

» Example Usage

```
data "openstack_networking_secgroup_v2" "secgroup" {
  name = "tf_test_secgroup"
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve security groups ids. If omitted, the `region` argument of the provider is used.
- `secgroup_id` - (Optional) The ID of the security group.
- `name` - (Optional) The name of the security group.
- `tenant_id` - (Optional) The owner of the security group.

» Attributes Reference

`id` is set to the ID of the found security group. In addition, the following attributes are exported:

- `name` - See Argument Reference above.
- `description`- The description of the security group.
- `region` - See Argument Reference above.

» openstack_networking_subnet_v2

Use this data source to get the ID of an available OpenStack subnet.

» Example Usage

```
data "openstack_networking_subnet_v2" "subnet_1" {
  name = "subnet_1"
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve subnet ids. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the subnet.
- `dhcp_enabled` - (Optional) If the subnet has DHCP enabled.
- `dhcp_disabled` - (Optional) If the subnet has DHCP disabled.
- `ip_version` - (Optional) The IP version of the subnet (either 4 or 6).
- `ipv6_address_mode` - (Optional) The IPv6 address mode. Valid values are `dhcpv6-stateful`, `dhcpv6-stateless`, or `slaac`.
- `ipv6_ra_mode` - (Optional) The IPv6 Router Advertisement mode. Valid values are `dhcpv6-stateful`, `dhcpv6-stateless`, or `slaac`.
- `gateway_ip` - (Optional) The IP of the subnet's gateway.
- `cidr` - (Optional) The CIDR of the subnet.
- `subnet_id` - (Optional) The ID of the subnet.
- `subnetpool_id` - (Optional) The ID of the subnetpool associated with the subnet.
- `network_id` - (Optional) The ID of the network the subnet belongs to.
- `tenant_id` - (Optional) The owner of the subnet.

» Attributes Reference

`id` is set to the ID of the found subnet. In addition, the following attributes are exported:

- `allocation_pools` - Allocation pools of the subnet.
- `enable_dhcp` - Whether the subnet has DHCP enabled or not.
- `dns_nameservers` - DNS Nameservers of the subnet.
- `host_routes` - Host Routes of the subnet.
- `region` - See Argument Reference above.

» openstack_networking_subnetpool_v2

Use this data source to get the ID of an available OpenStack subnetpool.

» Example Usage

```
data "openstack_networking_subnetpool_v2" "subnetpool_1" {
  name = "subnetpool_1"
}
```

» Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to retrieve a subnetpool id. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the subnetpool.
- `default_quota` - (Optional) The per-project quota on the prefix space that can be allocated from the subnetpool for project subnets.
- `project_id` - (Optional) The owner of the subnetpool.
- `prefixes` - (Optional) A list of subnet prefixes that are assigned to the subnetpool.
- `default_prefixlen` - (Optional) The size of the subnetpool default prefix length.
- `min_prefixlen` - (Optional) The size of the subnetpool min prefix length.
- `max_prefixlen` - (Optional) The size of the subnetpool max prefix length.
- `address_scope_id` - (Optional) The Neutron address scope that subnetpools is assigned to.
- `ip_version` - The IP protocol version.
- `shared` - (Optional) Whether this subnetpool is shared across all projects.
- `description` - (Optional) The human-readable description for the subnetpool.
- `is_default` - (Optional) Whether the subnetpool is default subnetpool or not.

» Attributes Reference

`id` is set to the ID of the found subnetpool. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `default_quota` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `created_at` - The time at which subnetpool was created.
- `updated_at` - The time at which subnetpool was created.
- `prefixes` - See Argument Reference above.
- `default_prefixlen` - See Argument Reference above.
- `min_prefixlen` - See Argument Reference above.
- `max_prefixlen` - See Argument Reference above.
- `address_scope_id` - See Argument Reference above.
- `ip_version` -The IP protocol version.
- `shared` - See Argument Reference above.
- `description` - See Argument Reference above.
- `is_default` - See Argument Reference above.
- `revision_number` - The revision number of the subnetpool.

» openstack_blockstorage_volume_v1

Manages a V1 volume resource within OpenStack.

» Example Usage

```
resource "openstack_blockstorage_volume_v1" "volume_1" {
  region      = "RegionOne"
  name        = "tf-test-volume"
  description = "first test volume"
  size        = 3
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to create the volume. If omitted, the `region` argument of the provider is used. Changing this creates a new volume.

- `size` - (Required) The size of the volume to create (in gigabytes). Changing this creates a new volume.
- `name` - (Optional) A unique name for the volume. Changing this updates the volume's name.
- `description` - (Optional) A description of the volume. Changing this updates the volume's description.
- `availability_zone` - (Optional) The availability zone for the volume. Changing this creates a new volume.
- `image_id` - (Optional) The image ID from which to create the volume. Changing this creates a new volume.
- `snapshot_id` - (Optional) The snapshot ID from which to create the volume. Changing this creates a new volume.
- `source_vol_id` - (Optional) The volume ID from which to create the volume. Changing this creates a new volume.
- `metadata` - (Optional) Metadata key/value pairs to associate with the volume. Changing this updates the existing volume metadata.
- `volume_type` - (Optional) The type of volume to create. Changing this creates a new volume.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `size` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `availability_zone` - See Argument Reference above.
- `image_id` - See Argument Reference above.
- `source_vol_id` - See Argument Reference above.
- `snapshot_id` - See Argument Reference above.
- `metadata` - See Argument Reference above.
- `volume_type` - See Argument Reference above.
- `attachment` - If a volume is attached to an instance, this attribute will display the Attachment ID, Instance ID, and the Device as the Instance sees it.

» Import

Volumes can be imported using the `id`, e.g.

```
$ terraform import openstack_blockstorage_volume_v1.volume_1 ea257959-eeb1-4c10-8d33-26f0409
```

» openstack_blockstorage_volume_v2

Manages a V2 volume resource within OpenStack.

» Example Usage

```
resource "openstack_blockstorage_volume_v2" "volume_1" {
  region      = "RegionOne"
  name        = "volume_1"
  description = "first test volume"
  size        = 3
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to create the volume. If omitted, the `region` argument of the provider is used. Changing this creates a new volume.
- `size` - (Required) The size of the volume to create (in gigabytes). Changing this creates a new volume.
- `availability_zone` - (Optional) The availability zone for the volume. Changing this creates a new volume.
- `consistency_group_id` - (Optional) The consistency group to place the volume in.
- `description` - (Optional) A description of the volume. Changing this updates the volume's description.
- `image_id` - (Optional) The image ID from which to create the volume. Changing this creates a new volume.
- `metadata` - (Optional) Metadata key/value pairs to associate with the volume. Changing this updates the existing volume metadata.
- `name` - (Optional) A unique name for the volume. Changing this updates the volume's name.
- `snapshot_id` - (Optional) The snapshot ID from which to create the volume. Changing this creates a new volume.

- `source_replica` - (Optional) The volume ID to replicate with.
- `source_vol_id` - (Optional) The volume ID from which to create the volume. Changing this creates a new volume.
- `volume_type` - (Optional) The type of volume to create. Changing this creates a new volume.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `size` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `availability_zone` - See Argument Reference above.
- `image_id` - See Argument Reference above.
- `source_vol_id` - See Argument Reference above.
- `snapshot_id` - See Argument Reference above.
- `metadata` - See Argument Reference above.
- `volume_type` - See Argument Reference above.
- `attachment` - If a volume is attached to an instance, this attribute will display the Attachment ID, Instance ID, and the Device as the Instance sees it.

» Import

Volumes can be imported using the `id`, e.g.

```
$ terraform import openstack_blockstorage_volume_v2.volume_1 ea257959-eeb1-4c10-8d33-26f0409
```

» openstack_blockstorage_volume_attach_v2

This resource is experimental and may be removed in the future! Feedback is requested if you find this resource useful or if you find any problems with it.

Creates a general purpose attachment connection to a Block Storage volume using the OpenStack Block Storage (Cinder) v2 API. Depending on your Block Storage service configuration, this resource can assist in attaching a volume to a non-OpenStack resource such as a bare-metal server or a remote virtual machine in a different cloud provider.

This does not actually attach a volume to an instance. Please use the `openstack_compute_volume_attach_v2` resource for that.

» Example Usage

```
resource "openstack_blockstorage_volume_v2" "volume_1" {
  name = "volume_1"
  size = 1
}

resource "openstack_blockstorage_volume_attach_v2" "va_1" {
  volume_id = "${openstack_blockstorage_volume_v2.volume_1.id}"
  device = "auto"
  host_name = "devstack"
  ip_address = "192.168.255.10"
  initiator = "iqn.1993-08.org.debian:01:e9861fb1859"
  os_type = "linux2"
  platform = "x86_64"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Block Storage client. A Block Storage client is needed to create a volume attachment. If omitted, the `region` argument of the provider is used. Changing this creates a new volume attachment.
- `attach_mode` - (Optional) Specify whether to attach the volume as Read-Only (`ro`) or Read-Write (`rw`). Only values of `ro` and `rw` are accepted. If left unspecified, the Block Storage API will apply a default of `rw`.
- `device` - (Optional) The device to tell the Block Storage service this volume will be attached as. This is purely for informational purposes. You can specify `auto` or a device such as `/dev/vdc`.
- `host_name` - (Required) The host to attach the volume to.
- `initiator` - (Optional) The iSCSI initiator string to make the connection.
- `ip_address` - (Optional) The IP address of the `host_name` above.
- `multipath` - (Optional) Whether to connect to this volume via multipath.
- `os_type` - (Optional) The iSCSI initiator OS type.
- `platform` - (Optional) The iSCSI initiator platform.
- `volume_id` - (Required) The ID of the Volume to attach to an Instance.
- `wwpn` - (Optional) An array of wwpn strings. Used for Fibre Channel connections.

- `wwnn` - (Optional) A wwnn name. Used for Fibre Channel connections.

» Attributes Reference

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

- `data` - This is a map of key/value pairs that contain the connection information. You will want to pass this information to a provisioner script to finalize the connection. See below for more information.
- `driver_volume_type` - The storage driver that the volume is based on.
- `mount_point_base` - A mount point base name for shared storage.

» Volume Connection Data

Upon creation of this resource, a `data` exported attribute will be available. This attribute is a set of key/value pairs that contains the information required to complete the block storage connection.

As an example, creating an iSCSI-based volume will return the following:

```
data.access_mode = rw
data.auth_method = CHAP
data.auth_password = xUhbGKQ8QCwKmHQ2
data.auth_username = Sphn5X4EoyFUUMYVYSA4
data.target_iqn = iqn.2010-10.org.openstack:volume-2d87ed25-c312-4f42-be1d-3b36b014561d
data.target_portal = 192.168.255.10:3260
data.volume_id = 2d87ed25-c312-4f42-be1d-3b36b014561d
```

This information can then be fed into a provisioner or a template shell script, where the final result would look something like:

```
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --interface defaul
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n noc
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n noc
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n noc
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --login
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n noc
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --rescan
```

The contents of `data` will vary from each Block Storage service. You must have a good understanding of how the service is configured and how to make the appropriate final connection. However, if used correctly, this has the flexibility to be able to attach OpenStack Block Storage volumes to non-OpenStack resources.

» Import

It is not possible to import this resource.

» openstack_compute_flavor_v2

Manages a V2 flavor resource within OpenStack.

» Example Usage

```
resource "openstack_compute_flavor_v2" "test-flavor" {
  name  = "my-flavor"
  ram   = "8096"
  vcpus = "2"
  disk  = "20"

  extra_specs {
    "hw:cpu_policy" = "CPU-POLICY",
    "hw:cpu_thread_policy" = "CPU-THREAD-POLICY"
  }
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. Flavors are associated with accounts, but a Compute client is needed to create one. If omitted, the `region` argument of the provider is used. Changing this creates a new flavor.
- `name` - (Required) A unique name for the flavor. Changing this creates a new flavor.
- `ram` - (Required) The amount of RAM to use, in megabytes. Changing this creates a new flavor.
- `vcpus` - (Required) The number of virtual CPUs to use. Changing this creates a new flavor.
- `disk` - (Required) The amount of disk space in gigabytes to use for the root (/) partition. Changing this creates a new flavor.
- `swap` - (Optional) The amount of disk space in megabytes to use. If unspecified, the default is 0. Changing this creates a new flavor.

- `rx_tx_factor` - (Optional) RX/TX bandwith factor. The default is 1. Changing this creates a new flavor.
- `is_public` - (Optional) Whether the flavor is public. Changing this creates a new flavor.
- `extra_specs` - (Optional) Key/Value pairs of metadata for the flavor.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `ram` - See Argument Reference above.
- `vcpus` - See Argument Reference above.
- `disk` - See Argument Reference above.
- `swap` - See Argument Reference above.
- `rx_tx_factor` - See Argument Reference above.
- `is_public` - See Argument Reference above.
- `extra_specs` - See Argument Reference above.

» Import

Flavors can be imported using the `ID`, e.g.

```
$ terraform import openstack_compute_flavor_v2.my-flavor 4142e64b-1b35-44a0-9b1e-5affc7af11(
```

» openstack_compute_floatingip_v2

Manages a V2 floating IP resource within OpenStack Nova (compute) that can be used for compute instances.

Please note that managing floating IPs through the OpenStack Compute API has been deprecated. Unless you are using an older OpenStack environment, it is recommended to use the `openstack_networking_floatingip_v2` resource instead, which uses the OpenStack Networking API.

» Example Usage

```
resource "openstack_compute_floatingip_v2" "floatip_1" {
  pool = "public"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. A Compute client is needed to create a floating IP that can be used with a compute instance. If omitted, the `region` argument of the provider is used. Changing this creates a new floating IP (which may or may not have a different address).
- `pool` - (Required) The name of the pool from which to obtain the floating IP. Changing this creates a new floating IP.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `pool` - See Argument Reference above.
- `address` - The actual floating IP address itself.
- `fixed_ip` - The fixed IP address corresponding to the floating IP.
- `instance_id` - UUID of the compute instance associated with the floating IP.

» Import

Floating IPs can be imported using the `id`, e.g.

```
$ terraform import openstack_compute_floatingip_v2.floatip_1 89c60255-9bd6-460c-822a-e2b959e
```

» openstack_compute_floatingip_associate_v2

Associate a floating IP to an instance. This can be used instead of the `floating_ip` options in `openstack_compute_instance_v2`.

» Example Usage

» Automatically detect the correct network

```
resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = 3
```

```
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]
}

resource "openstack_networking_floatingip_v2" "fip_1" {
  pool = "my_pool"
}

resource "openstack_compute_floatingip_associate_v2" "fip_1" {
  floating_ip = "${openstack_networking_floatingip_v2.fip_1.address}"
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
}
```

» Explicitly set the network to attach to

```
resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = 3
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  network {
    name = "my_network"
  }

  network {
    name = "default"
  }
}

resource "openstack_networking_floatingip_v2" "fip_1" {
  pool = "my_pool"
}

resource "openstack_compute_floatingip_associate_v2" "fip_1" {
  floating_ip = "${openstack_networking_floatingip_v2.fip_1.address}"
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  fixed_ip    = "${openstack_compute_instance_v2.instance_1.network.1.fixed_ip_v4}"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. Keypairs are associated with accounts, but a Compute client is needed to create one. If omitted, the `region` argument of the provider is used. Changing this creates a new floatingip_associate.
- `floating_ip` - (Required) The floating IP to associate.
- `instance_id` - (Required) The instance to associte the floating IP with.
- `fixed_ip` - (Optional) The specific IP address to direct traffic to.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `floating_ip` - See Argument Reference above.
- `instance_id` - See Argument Reference above.
- `fixed_ip` - See Argument Reference above.

» Import

This resource can be imported by specifying all three arguments, separated by a forward slash:

```
$ terraform import openstack_compute_floatingip_associate_v2.fip_1 <floating_ip>/<instance_i
```

» openstack_compute_instance_v2

Manages a V2 VM instance resource within OpenStack.

» Example Usage

» Basic Instance

```
resource "openstack_compute_instance_v2" "basic" {
  name            = "basic"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  metadata {
    this = "that"
```

```
  }

  network {
    name = "my_network"
  }
}
```

» Instance With Attached Volume

```
resource "openstack_blockstorage_volume_v2" "myvol" {
  name = "myvol"
  size = 1
}

resource "openstack_compute_instance_v2" "myinstance" {
  name            = "myinstance"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  network {
    name = "my_network"
  }
}

resource "openstack_compute_volume_attach_v2" "attached" {
  compute_id = "${openstack_compute_instance_v2.myinstance.id}"
  volume_id = "${openstack_blockstorage_volume_v2.myvol.id}"
}
```

» Boot From Volume

```
resource "openstack_compute_instance_v2" "boot-from-volume" {
  name            = "boot-from-volume"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  block_device {
    uuid                  = "<image-id>"
    source_type           = "image"
    volume_size           = 5
    boot_index            = 0
```

```
    destination_type      = "volume"
    delete_on_termination = true
  }

  network {
    name = "my_network"
  }
}
```

» Boot From an Existing Volume

```
resource "openstack_blockstorage_volume_v1" "myvol" {
  name     = "myvol"
  size     = 5
  image_id = "<image-id>"
}

resource "openstack_compute_instance_v2" "boot-from-volume" {
  name            = "bootfromvolume"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  block_device {
    uuid                  = "${openstack_blockstorage_volume_v1.myvol.id}"
    source_type           = "volume"
    boot_index            = 0
    destination_type      = "volume"
    delete_on_termination = true
  }

  network {
    name = "my_network"
  }
}
```

» Boot Instance, Create Volume, and Attach Volume as a Block Device

```
resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  image_id        = "<image-id>"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
```

```
  security_groups = ["default"]

  block_device {
    uuid                  = "<image-id>"
    source_type           = "image"
    destination_type      = "local"
    boot_index            = 0
    delete_on_termination = true
  }

  block_device {
    source_type           = "blank"
    destination_type      = "volume"
    volume_size           = 1
    boot_index            = 1
    delete_on_termination = true
  }
}
```

» Boot Instance and Attach Existing Volume as a Block Device

```
resource "openstack_blockstorage_volume_v2" "volume_1" {
  name = "volume_1"
  size = 1
}

resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  image_id        = "<image-id>"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  block_device {
    uuid                  = "<image-id>"
    source_type           = "image"
    destination_type      = "local"
    boot_index            = 0
    delete_on_termination = true
  }

  block_device {
    uuid                  = "${openstack_blockstorage_volume_v2.volume_1.id}"
    source_type           = "volume"
    destination_type      = "volume"
```

```
    boot_index            = 1
    delete_on_termination = true
  }
}
```

» Instance With Multiple Networks

```
resource "openstack_networking_floatingip_v2" "myip" {
  pool = "my_pool"
}

resource "openstack_compute_instance_v2" "multi-net" {
  name            = "multi-net"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  network {
    name = "my_first_network"
  }

  network {
    name = "my_second_network"
  }
}

resource "openstack_compute_floatingip_associate_v2" "myip" {
  floating_ip = "${openstack_networking_floatingip_v2.myip.address}"
  instance_id = "${openstack_compute_instance_v2.multi-net.id}"
  fixed_ip = "${openstack_compute_instance_v2.multi-net.network.1.fixed_ip_v4}"
}
```

» Instance With Personality

```
resource "openstack_compute_instance_v2" "personality" {
  name            = "personality"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  personality {
    file    = "/path/to/file/on/instance.txt"
```

```
    content = "contents of file"
  }

  network {
    name = "my_network"
  }
}
```

» Instance with Multiple Ephemeral Disks

```
resource "openstack_compute_instance_v2" "multi-eph" {
  name            = "multi_eph"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]

  block_device {
    boot_index            = 0
    delete_on_termination = true
    destination_type      = "local"
    source_type           = "image"
    uuid                  = "<image-id>"
  }

  block_device {
    boot_index            = -1
    delete_on_termination = true
    destination_type      = "local"
    source_type           = "blank"
    volume_size           = 1
  }

  block_device {
    boot_index            = -1
    delete_on_termination = true
    destination_type      = "local"
    source_type           = "blank"
    volume_size           = 1
  }
}
```

» Instance with User Data (cloud-init)

```
resource "openstack_compute_instance_v2" "instance_1" {
  name            = "basic"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
  key_pair        = "my_key_pair_name"
  security_groups = ["default"]
  user_data       = "#cloud-config\nhostname: instance_1.example.com\nfqdn: instance_1.examp

  network {
    name = "my_network"
  }
}
```

`user_data` can come from a variety of sources: inline, read in from the `file` function, or the `template_cloudinit_config` resource.

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to create the server instance. If omitted, the `region` argument of the provider is used. Changing this creates a new server.
- `name` - (Required) A unique name for the resource.
- `image_id` - (Optional; Required if `image_name` is empty and not booting from a volume. Do not specify if booting from a volume.) The image ID of the desired image for the server. Changing this creates a new server.
- `image_name` - (Optional; Required if `image_id` is empty and not booting from a volume. Do not specify if booting from a volume.) The name of the desired image for the server. Changing this creates a new server.
- `flavor_id` - (Optional; Required if `flavor_name` is empty) The flavor ID of the desired flavor for the server. Changing this resizes the existing server.
- `flavor_name` - (Optional; Required if `flavor_id` is empty) The name of the desired flavor for the server. Changing this resizes the existing server.
- `user_data` - (Optional) The user data to provide when launching the instance. Changing this creates a new server.
- `security_groups` - (Optional) An array of one or more security group names to associate with the server. Changing this results in adding/removing security groups from the existing server. *Note*: When

attaching the instance to networks using Ports, place the security groups on the Port and not the instance.

- `availability_zone` - (Optional) The availability zone in which to create the server. Changing this creates a new server.
- `network` - (Optional) An array of one or more networks to attach to the instance. The network object structure is documented below. Changing this creates a new server.
- `metadata` - (Optional) Metadata key/value pairs to make available from within the instance. Changing this updates the existing server metadata.
- `config_drive` - (Optional) Whether to use the config_drive feature to configure the instance. Changing this creates a new server.
- `admin_pass` - (Optional) The administrative password to assign to the server. Changing this changes the root password on the existing server.
- `key_pair` - (Optional) The name of a key pair to put on the server. The key pair must already be created and associated with the tenant's account. Changing this creates a new server.
- `block_device` - (Optional) Configuration of block devices. The block_device structure is documented below. Changing this creates a new server. You can specify multiple block devices which will create an instance with multiple disks. This configuration is very flexible, so please see the following reference for more information.
- `scheduler_hints` - (Optional) Provide the Nova scheduler with hints on how the instance should be launched. The available hints are described below.
- `personality` - (Optional) Customize the personality of an instance by defining one or more files and their contents. The personality structure is described below.
- `stop_before_destroy` - (Optional) Whether to try stop instance gracefully before destroying it, thus giving chance for guest OS daemons to stop correctly. If instance doesn't stop within timeout, it will be destroyed anyway.
- `force_delete` - (Optional) Whether to force the OpenStack instance to be forcefully deleted. This is useful for environments that have reclaim / soft deletion enabled.

The `network` block supports:

- `uuid` - (Required unless `port` or `name` is provided) The network UUID to attach to the server. Changing this creates a new server.
- `name` - (Required unless `uuid` or `port` is provided) The human-readable name of the network. Changing this creates a new server.

- `port` - (Required unless `uuid` or `name` is provided) The port UUID of a network to attach to the server. Changing this creates a new server.
- `fixed_ip_v4` - (Optional) Specifies a fixed IPv4 address to be used on this network. Changing this creates a new server.
- `fixed_ip_v6` - (Optional) Specifies a fixed IPv6 address to be used on this network. Changing this creates a new server.
- `access_network` - (Optional) Specifies if this network should be used for provisioning access. Accepts true or false. Defaults to false.

The `block_device` block supports:

- `uuid` - (Required unless `source_type` is set to `"blank"`) The UUID of the image, volume, or snapshot. Changing this creates a new server.
- `source_type` - (Required) The source type of the device. Must be one of "blank", "image", "volume", or "snapshot". Changing this creates a new server.
- `volume_size` - The size of the volume to create (in gigabytes). Required in the following combinations: source=image and destination=volume, source=blank and destination=local, and source=blank and destination=volume. Changing this creates a new server.
- `boot_index` - (Optional) The boot index of the volume. It defaults to 0. Changing this creates a new server.
- `destination_type` - (Optional) The type that gets created. Possible values are "volume" and "local". Changing this creates a new server.
- `delete_on_termination` - (Optional) Delete the volume / block device upon termination of the instance. Defaults to false. Changing this creates a new server.

The `scheduler_hints` block supports:

- `group` - (Optional) A UUID of a Server Group. The instance will be placed into that group.
- `different_host` - (Optional) A list of instance UUIDs. The instance will be scheduled on a different host than all other instances.
- `same_host` - (Optional) A list of instance UUIDs. The instance will be scheduled on the same host of those specified.
- `query` - (Optional) A conditional query that a compute node must pass in order to host an instance.
- `target_cell` - (Optional) The name of a cell to host the instance.
- `build_near_host_ip` - (Optional) An IP Address in CIDR form. The instance will be placed on a compute node that is in the same subnet.

- `additional_properties` - (Optional) Arbitrary key/value pairs of additional properties to pass to the scheduler.

The `personality` block supports:

- `file` - (Required) The absolute path of the destination file.
- `contents` - (Required) The contents of the file. Limited to 255 bytes.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `access_ip_v4` - The first detected Fixed IPv4 address *or* the Floating IP.
- `access_ip_v6` - The first detected Fixed IPv6 address.
- `metadata` - See Argument Reference above.
- `security_groups` - See Argument Reference above.
- `flavor_id` - See Argument Reference above.
- `flavor_name` - See Argument Reference above.
- `network/uuid` - See Argument Reference above.
- `network/name` - See Argument Reference above.
- `network/port` - See Argument Reference above.
- `network/fixed_ip_v4` - The Fixed IPv4 address of the Instance on that network.
- `network/fixed_ip_v6` - The Fixed IPv6 address of the Instance on that network.
- `network/mac` - The MAC address of the NIC on that network.
- `all_metadata` - Contains all instance metadata, even metadata not set by Terraform.

» Notes

» Multiple Ephemeral Disks

It's possible to specify multiple `block_device` entries to create an instance with multiple ephemeral (local) disks. In order to create multiple ephemeral disks, the sum of the total amount of ephemeral space must be less than or equal to what the chosen flavor supports.

The following example shows how to create an instance with multiple ephemeral disks:

```
resource "openstack_compute_instance_v2" "foo" {
  name            = "terraform-test"
  security_groups = ["default"]
```

```
  block_device {
    boot_index            = 0
    delete_on_termination = true
    destination_type      = "local"
    source_type           = "image"
    uuid                  = "<image uuid>"
  }

  block_device {
    boot_index            = -1
    delete_on_termination = true
    destination_type      = "local"
    source_type           = "blank"
    volume_size           = 1
  }

  block_device {
    boot_index            = -1
    delete_on_termination = true
    destination_type      = "local"
    source_type           = "blank"
    volume_size           = 1
  }
}
```

» Instances and Security Groups

When referencing a security group resource in an instance resource, always use the *name* of the security group. If you specify the ID of the security group, Terraform will remove and reapply the security group upon each call. This is because the OpenStack Compute API returns the names of the associated security groups and not their IDs.

Note the following example:

```
resource "openstack_networking_secgroup_v2" "sg_1" {
  name = "sg_1"
}

resource "openstack_compute_instance_v2" "foo" {
  name            = "terraform-test"
  security_groups = ["${openstack_networking_secgroup_v2.sg_1.name}"]
}
```

» Instances and Ports

Neutron Ports are a great feature and provide a lot of functionality. However, there are some notes to be aware of when mixing Instances and Ports:

- In OpenStack environments prior to the Kilo release, deleting or recreating an Instance will cause the Instance's Port(s) to be deleted. One way of working around this is to taint any Port(s) used in Instances which are to be recreated. See here for further information.
- When attaching an Instance to one or more networks using Ports, place the security groups on the Port and not the Instance. If you place the security groups on the Instance, the security groups will not be applied upon creation, but they will be applied upon a refresh. This is a known OpenStack bug.
- Network IP information is not available within an instance for networks that are attached with Ports. This is mostly due to the flexibility Neutron Ports provide when it comes to IP addresses. For example, a Neutron Port can have multiple Fixed IP addresses associated with it. It's not possible to know which single IP address the user would want returned to the Instance's state information. Therefore, in order for a Provisioner to connect to an Instance via it's network Port, customize the `connection` information:

```
resource "openstack_networking_port_v2" "port_1" {
  name           = "port_1"
  admin_state_up = "true"

  network_id = "0a1d0a27-cffa-4de3-92c5-9d3fd3f2e74d"

  security_group_ids = [
    "2f02d20a-8dca-49b7-b26f-b6ce9fddaf4f",
    "ca1e5ed7-dae8-4605-987b-fadaeeb30461",
  ]
}

resource "openstack_compute_instance_v2" "instance_1" {
  name = "instance_1"

  network {
    port = "${openstack_networking_port_v2.port_1.id}"
  }

  connection {
    user        = "root"
    host        = "${openstack_networking_port_v2.port_1.fixed_ip.0.ip_address}"
```

```
    private_key = "~/path/to/key"
  }

  provisioner "remote-exec" {
    inline = [
      "echo terraform executed > /tmp/foo",
    ]
  }
}
```

» openstack_compute_keypair_v2

Manages a V2 keypair resource within OpenStack.

Important Security Notice The private key generated by this resource will be stored *unencrypted* in your Terraform state file. **Use of this resource for production deployments is *not* recommended**. Instead, generate a private key file outside of Terraform and distribute it securely to the system where Terraform will be run.

» Example Usage

» Import an Existing Public Key

```
resource "openstack_compute_keypair_v2" "test-keypair" {
  name       = "my-keypair"
  public_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDAjpC1hwiOCCmKEWxJ4qzTTsJbKzndLotBCz5
}
```

» Generate a Public/Private Key Pair

```
resource "openstack_compute_keypair_v2" "test-keypair" {
  name = "my-keypair"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. Keypairs are associated with accounts, but a Compute client is needed to create one. If omitted, the `region` argument of the provider is used. Changing this creates a new keypair.

- `name` - (Required) A unique name for the keypair. Changing this creates a new keypair.
- `public_key` - (Optional) A pregenerated OpenSSH-formatted public key. Changing this creates a new keypair. If a public key is not specified, then a public/private key pair will be automatically generated. If a pair is created, then destroying this resource means you will lose access to that keypair forever.
- `value_specs` - (Optional) Map of additional options.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `public_key` - See Argument Reference above.
- `fingerprint` - The fingerprint of the public key.
- `private_key` - The generated private key when no public key is specified.

» Import

Keypairs can be imported using the `name`, e.g.

```
$ terraform import openstack_compute_keypair_v2.my-keypair test-keypair
```

» openstack_compute_secgroup_v2

Manages a V2 security group resource within OpenStack.

Please note that managing security groups through the OpenStack Compute API has been deprecated. Unless you are using an older OpenStack environment, it is recommended to use the `openstack_networking_secgroup_v2` and `openstack_networking_secgroup_rule_v2` resources instead, which uses the OpenStack Networking API.

» Example Usage

```
resource "openstack_compute_secgroup_v2" "secgroup_1" {
  name        = "my_secgroup"
  description = "my security group"

  rule {
```

```
    from_port   = 22
    to_port     = 22
    ip_protocol = "tcp"
    cidr        = "0.0.0.0/0"
  }

  rule {
    from_port   = 80
    to_port     = 80
    ip_protocol = "tcp"
    cidr        = "0.0.0.0/0"
  }
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. A Compute client is needed to create a security group. If omitted, the `region` argument of the provider is used. Changing this creates a new security group.
- `name` - (Required) A unique name for the security group. Changing this updates the `name` of an existing security group.
- `description` - (Required) A description for the security group. Changing this updates the `description` of an existing security group.
- `rule` - (Optional) A rule describing how the security group operates. The rule object structure is documented below. Changing this updates the security group rules. As shown in the example above, multiple rule blocks may be used.

The `rule` block supports:

- `from_port` - (Required) An integer representing the lower bound of the port range to open. Changing this creates a new security group rule.
- `to_port` - (Required) An integer representing the upper bound of the port range to open. Changing this creates a new security group rule.
- `ip_protocol` - (Required) The protocol type that will be allowed. Changing this creates a new security group rule.
- `cidr` - (Optional) Required if `from_group_id` or `self` is empty. The IP range that will be the source of network traffic to the security group. Use 0.0.0.0/0 to allow all IP addresses. Changing this creates a new security group rule. Cannot be combined with `from_group_id` or `self`.

- `from_group_id` - (Optional) Required if `cidr` or `self` is empty. The ID of a group from which to forward traffic to the parent group. Changing this creates a new security group rule. Cannot be combined with `cidr` or `self`.
- `self` - (Optional) Required if `cidr` and `from_group_id` is empty. If true, the security group itself will be added as a source to this ingress rule. Cannot be combined with `cidr` or `from_group_id`.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `rule` - See Argument Reference above.

» Notes

» ICMP Rules

When using ICMP as the `ip_protocol`, the `from_port` sets the ICMP *type* and the `to_port` sets the ICMP *code*. To allow all ICMP types, set each value to `-1`, like so:

```
rule {
  from_port = -1
  to_port = -1
  ip_protocol = "icmp"
  cidr = "0.0.0.0/0"
}
```

A list of ICMP types and codes can be found here.

» Referencing Security Groups

When referencing a security group in a configuration (for example, a configuration creates a new security group and then needs to apply it to an instance being created in the same configuration), it is currently recommended to reference the security group by name and not by ID, like this:

```
resource "openstack_compute_instance_v2" "test-server" {
  name            = "tf-test"
  image_id        = "ad091b52-742f-469e-8f3c-fd81cadf0743"
  flavor_id       = "3"
```

```
  key_pair        = "my_key_pair_name"
  security_groups = ["${openstack_compute_secgroup_v2.secgroup_1.name}"]
}
```

» Import

Security Groups can be imported using the `id`, e.g.

```
$ terraform import openstack_compute_secgroup_v2.my_secgroup 1bc30ee9-9d5b-4c30-bdd5-7f1e663
```

» openstack_compute_servergroup_v2

Manages a V2 Server Group resource within OpenStack.

» Example Usage

```
resource "openstack_compute_servergroup_v2" "test-sg" {
  name     = "my-sg"
  policies = ["anti-affinity"]
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. If omitted, the `region` argument of the provider is used. Changing this creates a new server group.
- `name` - (Required) A unique name for the server group. Changing this creates a new server group.
- `policies` - (Required) The set of policies for the server group. Only two two policies are available right now, and both are mutually exclusive. See the Policies section for more information. Changing this creates a new server group.
- `value_specs` - (Optional) Map of additional options.

» Policies

- `affinity` - All instances/servers launched in this group will be hosted on the same compute node.

- `anti-affinity` - All instances/servers launched in this group will be hosted on different compute nodes.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `policies` - See Argument Reference above.
- `members` - The instances that are part of this server group.

» Import

Server Groups can be imported using the `id`, e.g.

```
$ terraform import openstack_compute_servergroup_v2.test-sg 1bc30ee9-9d5b-4c30-bdd5-7f1e663
```

» openstack_compute_volume_attach_v2

Attaches a Block Storage Volume to an Instance using the OpenStack Compute (Nova) v2 API.

» Example Usage

» Basic attachment of a single volume to a single instance

```
resource "openstack_blockstorage_volume_v2" "volume_1" {
  name = "volume_1"
  size = 1
}

resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  security_groups = ["default"]
}

resource "openstack_compute_volume_attach_v2" "va_1" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  volume_id   = "${openstack_blockstorage_volume_v2.volume_1.id}"
}
```

» Attaching multiple volumes to a single instance

```
resource "openstack_blockstorage_volume_v2" "volumes" {
  count = 2
  name  = "${format("vol-%02d", count.index + 1)}"
  size  = 1
}

resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  security_groups = ["default"]
}

resource "openstack_compute_volume_attach_v2" "attachments" {
  count       = 2
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  volume_id   = "${element(openstack_blockstorage_volume_v2.volumes.*.id, count.index)}"
}

output "volume devices" {
  value = "${openstack_compute_volume_attach_v2.attachments.*.device}"
}
```

Note that the above example will not guarantee that the volumes are attached in a deterministic manner. The volumes will be attached in a seemingly random order.

If you want to ensure that the volumes are attached in a given order, create explicit dependencies between the volumes, such as:

```
resource "openstack_blockstorage_volume_v2" "volumes" {
  count = 2
  name  = "${format("vol-%02d", count.index + 1)}"
  size  = 1
}

resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  security_groups = ["default"]
}

resource "openstack_compute_volume_attach_v2" "attach_1" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  volume_id   = "${openstack_blockstorage_volume_v2.volumes.0.id}"
}

resource "openstack_compute_volume_attach_v2" "attach_2" {
```

```
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  volume_id   = "${openstack_blockstorage_volume_v2.volumes.1.id}"

  depends_on  = ["openstack_compute_volume_attach_v2.attach_1"]
}

output "volume devices" {
  value = "${openstack_compute_volume_attach_v2.attachments.*.device}"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. A Compute client is needed to create a volume attachment. If omitted, the `region` argument of the provider is used. Changing this creates a new volume attachment.
- `instance_id` - (Required) The ID of the Instance to attach the Volume to.
- `volume_id` - (Required) The ID of the Volume to attach to an Instance.
- `device` - (Optional) The device of the volume attachment (ex: `/dev/vdc`). *NOTE*: Being able to specify a device is dependent upon the hypervisor in use. There is a chance that the device specified in Terraform will not be the same device the hypervisor chose. If this happens, Terraform will wish to update the device upon subsequent applying which will cause the volume to be detached and reattached indefinitely. Please use with caution.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `instance_id` - See Argument Reference above.
- `volume_id` - See Argument Reference above.
- `device` - See Argument Reference above. *NOTE*: The correctness of this information is dependent upon the hypervisor in use. In some cases, this should not be used as an authoritative piece of information.

» Import

Volume Attachments can be imported using the Instance ID and Volume ID separated by a slash, e.g.

```
$ terraform import openstack_compute_volume_attach_v2.va_1 89c60255-9bd6-460c-822a-e2b959ede
```

» openstack_db_instance_v1

Manages a V1 DB instance resource within OpenStack.

» Example Usage

» Instance

```
resource "openstack_db_instance_v1" "test" {
  region    = "region-test"
  name      = "test"
  flavor_id = "31792d21-c355-4587-9290-56c1ed0ca376"
  size      = 8

  network {
    uuid = "c0612505-caf2-4fb0-b7cb-56a0240a2b12"
  }

  datastore {
    version = "mysql-5.7"
    type    = "mysql"
  }
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Required) The region in which to create the db instance. Changing this creates a new instance.
- `name` - (Required) A unique name for the resource.
- `flavor_id` - (Required) The flavor ID of the desired flavor for the instance. Changing this creates new instance.
- `configuration_id` - (Optional) Configuration ID to be attached to the instance. Database instance will be rebooted when configuration is detached.
- `size` - (Required) Specifies the volume size in GB. Changing this creates new instance.

- `datastore` - (Required) An array of database engine type and version. The datastore object structure is documented below. Changing this creates a new instance.
- `network` - (Optional) An array of one or more networks to attach to the instance. The network object structure is documented below. Changing this creates a new instance.
- `user` - (Optional) An array of username, password, host and databases. The user object structure is documented below.
- `database` - (Optional) An array of database name, charset and collate. The database object structure is documented below.

The `datastore` block supports:

- `type` - (Required) Database engine type to be used in new instance. Changing this creates a new instance.
- `version` - (Required) Version of database engine type to be used in new instance. Changing this creates a new instance.

The `network` block supports:

- `uuid` - (Required unless `port` is provided) The network UUID to attach to the instance. Changing this creates a new instance.
- `port` - (Required unless `uuid` is provided) The port UUID of a network to attach to the instance. Changing this creates a new instance.
- `fixed_ip_v4` - (Optional) Specifies a fixed IPv4 address to be used on this network. Changing this creates a new instance.
- `fixed_ip_v6` - (Optional) Specifies a fixed IPv6 address to be used on this network. Changing this creates a new instance.

The `user` block supports:

- `name` - (Optional) Username to be created on new instance. Changing this creates a new instance.
- `password` - (Optional) User's password. Changing this creates a new instance.
- `host` - (Optional) An ip address or % sign indicating what ip addresses can connect with this user credentials. Changing this creates a new instance.
- `databases` - (Optional) A list of databases that user will have access to. If not specified, user has access to all databases on th einstance. Changing this creates a new instance.

The `database` block supports:

- `name` - (Optional) Database to be created on new instance. Changing this creates a new instance.

- `collate` - (Optional) Database collation. Changing this creates a new instance.
- `charset` - (Optional) Database character set. Changing this creates a new instance.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `size` - See Argument Reference above.
- `flavor_id` - See Argument Reference above.
- `configuration_id` - See Argument Reference above.
- `datastore/type` - See Argument Reference above.
- `datastore/version` - See Argument Reference above.
- `network/uuid` - See Argument Reference above.
- `network/port` - See Argument Reference above.
- `network/fixed_ip_v4` - The Fixed IPv4 address of the Instance on that network.
- `network/fixed_ip_v6` - The Fixed IPv6 address of the Instance on that
- `database/name` - See Argument Reference above.
- `database/collate` - See Argument Reference above.
- `database/charset` - See Argument Reference above.
- `user/name` - See Argument Reference above.
- `user/password` - See Argument Reference above.
- `user/databases` - See Argument Reference above.
- `user/host` - See Argument Reference above.

» openstack_db_database_v1

Manages a V1 DB database resource within OpenStack.

» Example Usage

» Database

```
resource "openstack_db_database_v1" "mydb" {
  name        = "mydb"
  instance_id = "${openstack_db_instance_v1.basic.id}"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) A unique name for the resource.
- `instance_id` - (Required) The ID for the database instance.

» Attributes Reference

The following attributes are exported:

- `region` - Openstack region resource is created in.
- `name` - See Argument Reference above.
- `instance_id` - See Argument Reference above.

» Import

Databases can be imported by using `instance-id/db-name`, e.g.

```
$ terraform import openstack_db_database_v1.mydb 7b9e3cd3-00d9-449c-b074-8439f8e274fa/mydb
```

» openstack_db_user_v1

Manages a V1 DB user resource within OpenStack.

» Example Usage

» User

```
resource "openstack_db_user_v1" "basic" {
  name      = "basic"
  instance  = "${openstack_db_instance_v1.basic.id}"
  password  = "password"
  databases = ["testdb"]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) A unique name for the resource.

- `instance` - (Required) The ID for the database instance.
- `password` - (Required) User's password.
- `databases` - (Optional) A list of database user should have access to.

» Attributes Reference

The following attributes are exported:

- `region` - Openstack region resource is created in.
- `name` - See Argument Reference above.
- `instance` - See Argument Reference above.
- `password` - See Argument Reference above.
- `databases` - See Argument Reference above.

» openstack_db_configuration_v1

Manages a V1 DB configuration resource within OpenStack.

» Example Usage

» Configuration

```
resource "openstack_db_configuration_v1" "test" {
  name        = "test"
  description = "description"

  datastore {
    version = "mysql-5.7"
    type    = "mysql"
  }

  configuration {
    name  = "max_connections"
    value = 200
  }
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Required) The region in which to create the db instance. Changing this creates a new instance.
- `name` - (Required) A unique name for the resource.
- `description` - (Optional) Description of the resource.
- `datastore` - (Required) An array of database engine type and version. The datastore object structure is documented below. Changing this creates resource.
- `configuration` - (Optional) An array of configuration parameter name and value. Can be specified multiple times. The configuration object structure is documented below.

The `datastore` block supports:

- `type` - (Required) Database engine type to be used with this configuration. Changing this creates a new resource.
- `version` - (Required) Version of database engine type to be used with this configuration. Changing this creates a new resource.

The `configuration` block supports:

- `name` - (Optional) Configuration parameter name. Changing this creates a new resource.
- `value` - (Optional) Configuration parameter value. Changing this creates a new resource.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `datastore/type` - See Argument Reference above.
- `datastore/version` - See Argument Reference above.
- `configuration/name` - See Argument Reference above.
- `configuration/value` - See Argument Reference above.

» openstack_dns_recordset_v2

Manages a DNS record set in the OpenStack DNS Service.

» Example Usage

» Automatically detect the correct network

```
resource "openstack_dns_zone_v2" "example_zone" {
  name = "example.com."
  email = "email2@example.com"
  description = "a zone"
  ttl = 6000
  type = "PRIMARY"
}

resource "openstack_dns_recordset_v2" "rs_example_com" {
  zone_id = "${openstack_dns_zone_v2.example_zone.id}"
  name = "rs.example.com."
  description = "An example record set"
  ttl = 3000
  type = "A"
  records = ["10.0.0.1"]
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 DNS client. If omitted, the `region` argument of the provider is used. Changing this creates a new DNS record set.
- `zone_id` - (Required) The ID of the zone in which to create the record set. Changing this creates a new DNS record set.
- `name` - (Required) The name of the record set. Note the `.` at the end of the name. Changing this creates a new DNS record set.
- `type` - (Optional) The type of record set. Examples: "A", "MX". Changing this creates a new DNS record set.
- `ttl` - (Optional) The time to live (TTL) of the record set.
- `description` - (Optional) A description of the record set.
- `records` - (Optional) An array of DNS records.
- `value_specs` - (Optional) Map of additional options. Changing this creates a new record set.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `type` - See Argument Reference above.
- `ttl` - See Argument Reference above.
- `description` - See Argument Reference above.
- `records` - See Argument Reference above.
- `zone_id` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

» Import

This resource can be imported by specifying the zone ID and recordset ID, separated by a forward slash.

```
$ terraform import openstack_dns_recordset_v2.recordset_1 <zone_id>/<recordset_id>
```

» openstack_dns_zone_v2

Manages a DNS zone in the OpenStack DNS Service.

» Example Usage

» Automatically detect the correct network

```
resource "openstack_dns_zone_v2" "example.com" {
  name = "example.com."
  email = "jdoe@example.com"
  description = "An example zone"
  ttl = 3000
  type = "PRIMARY"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Compute client. Keypairs are associated with accounts, but a Compute client is needed

to create one. If omitted, the `region` argument of the provider is used. Changing this creates a new DNS zone.

- `name` - (Required) The name of the zone. Note the `.` at the end of the name. Changing this creates a new DNS zone.
- `email` - (Optional) The email contact for the zone record.
- `type` - (Optional) The type of zone. Can either be `PRIMARY` or `SECONDARY`. Changing this creates a new zone.
- `attributes` - (Optional) Attributes for the DNS Service scheduler. Changing this creates a new zone.
- `ttl` - (Optional) The time to live (TTL) of the zone.
- `description` - (Optional) A description of the zone.
- `masters` - (Optional) An array of master DNS servers. For when `type` is `SECONDARY`.
- `value_specs` - (Optional) Map of additional options. Changing this creates a new zone.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `email` - See Argument Reference above.
- `type` - See Argument Reference above.
- `attributes` - See Argument Reference above.
- `ttl` - See Argument Reference above.
- `description` - See Argument Reference above.
- `masters` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

» Import

This resource can be imported by specifying the zone ID:

```
$ terraform import openstack_dns_zone_v2.zone_1 <zone_id>
```

» openstack_identity_project_v3

Manages a V3 Project resource within OpenStack Keystone.

Note: You *must* have admin privileges in your OpenStack cloud to use this resource.

» Example Usage

```
resource "openstack_identity_project_v3" "project_1" {
  name = "project_1"
  description = "A project"
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) A description of the project.
- `domain_id` - (Optional) The domain this project belongs to.
- `enabled` - (Optional) Whether the project is enabled or disabled. Valid values are `true` and `false`.
- `is_domain` - (Optional) Whether this project is a domain. Valid values are `true` and `false`.
- `name` - (Optional) The name of the project.
- `parent_id` - (Optional) The parent of this project.
- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used. Changing this creates a new User.

» Attributes Reference

The following attributes are exported:

- `domain_id` - See Argument Reference above.
- `parent_id` - See Argument Reference above.

» Import

Projects can be imported using the `id`, e.g.

```
$ terraform import openstack_identity_project_v3.project_1 89c60255-9bd6-460c-822a-e2b959ede
```

» openstack_identity_role_v3

Manages a V3 Role resource within OpenStack Keystone.

Note: You *must* have admin privileges in your OpenStack cloud to use this resource.

» Example Usage

```
resource "openstack_identity_role_v3" "role_1" {
  name = "role_1"
}
```

» Argument Reference

The following arguments are supported:

- `name` - The name of the role.
- `domain_id` - (Optional) The domain the role belongs to.
- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used. Changing this creates a new Role.

» Attributes Reference

The following attributes are exported:

- `name` - See Argument Reference above.
- `domain_id` - See Argument Reference above.
- `region` - See Argument Reference above.

» Import

Roles can be imported using the `id`, e.g.

```
$ terraform import openstack_identity_role_v3.role_1 89c60255-9bd6-460c-822a-e2b959ede9d2
```

» openstack_identity_role_assignment_v3

Manages a V3 Role assignment within OpenStack Keystone.

Note: You *must* have admin privileges in your OpenStack cloud to use this resource.

» Example Usage

```
resource "openstack_identity_project_v3" "project_1" {
  name = "project_1"
}

resource "openstack_identity_user_v3" "user_1" {
  name = "user_1"
  default_project_id = "${openstack_identity_project_v3.project_1.id}"
}

resource "openstack_identity_role_v3" "role_1" {
  name = "role_1"
}

resource "openstack_identity_role_assignment_v3" "role_assignment_1" {
  user_id = "${openstack_identity_user_v3.user_1.id}"
  project_id = "${openstack_identity_project_v3.project_1.id}"
  role_id = "${openstack_identity_role_v3.role_1.id}"
}
```

» Argument Reference

The following arguments are supported:

- `domain_id` - (Optional; Required if `project_id` is empty) The domain to assign the role in.
- `group_id` - (Optional; Required if `user_id` is empty) The group to assign the role to.
- `project_id` - (Optional; Required if `domain_id` is empty) The project to assign the role in.
- `user_id` - (Optional; Required if `group_id` is empty) The user to assign the role to.
- `role_id` - (Required) The role to assign.

» Attributes Reference

The following attributes are exported:

- `domain_id` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `group_id` - See Argument Reference above.
- `user_id` - See Argument Reference above.
- `role_id` - See Argument Reference above.

» openstack_identity_user_v3

Manages a V3 User resource within OpenStack Keystone.

Note: You *must* have admin privileges in your OpenStack cloud to use this resource.

» Example Usage

```
resource "openstack_identity_project_v3" "project_1" {
  name = "project_1"
}

resource "openstack_identity_user_v3" "user_1" {
  default_project_id = "${openstack_identity_project_v3.project_1.id}"
  name = "user_1"
  description = "A user"

  password = "password123"

  ignore_change_password_upon_first_use = true

  multi_factor_auth_enabled = true

  multi_factor_auth_rule {
    rule = ["password", "totp"]
  }

  multi_factor_auth_rule {
    rule = ["password"]
  }

  extra {
    email = "user_1@foobar.com"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) A description of the user.
- `default_project_id` - (Optional) The default project this user belongs to.
- `domain_id` - (Optional) The domain this user belongs to.
- `enabled` - (Optional) Whether the user is enabled or disabled. Valid values are `true` and `false`.
- `extra` - (Optional) Free-form key/value pairs of extra information.
- `ignore_change_password_upon_first_use` - (Optional) User will not have to change their password upon first use. Valid values are `true` and `false`.
- `ignore_password_expiry` - (Optional) User's password will not expire. Valid values are `true` and `false`.
- `ignore_lockout_failure_attempts` - (Optional) User will not have a failure lockout placed on their account. Valid values are `true` and `false`.
- `multi_factor_auth_enabled` - (Optional) Whether to enable multi-factor authentication. Valid values are `true` and `false`.
- `multi_factor_auth_rule` - (Optional) A multi-factor authentication rule. The structure is documented below. Please see the Ocata release notes for more information on how to use mulit-factor rules.
- `name` - (Optional) The name of the user.
- `password` - (Optional) The password for the user.
- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used. Changing this creates a new User.

The `multi_factor_auth_rule` block supports:

- `rule` - (Required) A list of authentication plugins that the user must authenticate with.

» Attributes Reference

The following attributes are exported:

- `domain_id` - See Argument Reference above.

» Import

Users can be imported using the `id`, e.g.

```
$ terraform import openstack_identity_user_v3.user_1 89c60255-9bd6-460c-822a-e2b959ede9d2
```

» openstack_images_image_v2

Manages a V2 Image resource within OpenStack Glance.

» Example Usage

```
resource "openstack_images_image_v2" "rancheros" {
  name   = "RancherOS"
  image_source_url = "https://releases.rancher.com/os/latest/rancheros-openstack.img"
  container_format = "bare"
  disk_format = "qcow2"

  properties {
    key = "value"
  }
}
```

» Argument Reference

The following arguments are supported:

- `container_format` - (Required) The container format. Must be one of ”ami”, ”ari”, ”aki”, ”bare”, ”ovf”.
- `disk_format` - (Required) The disk format. Must be one of ”ami”, ”ari”, ”aki”, ”vhd”, ”vmdk”, ”raw”, ”qcow2”, ”vdi”, ”iso”.
- `local_file_path` - (Optional) This is the filepath of the raw image file that will be uploaded to Glance. Conflicts with `image_source_url`.
- `image_cache_path` - (Optional) This is the directory where the images will be downloaded. Images will be stored with a filename corresponding to the url’s md5 hash. Defaults to ”$HOME/.terraform/image_cache”
- `image_source_url` - (Optional) This is the url of the raw image that will be downloaded in the `image_cache_path` before being uploaded to Glance. Glance is able to download image from internet but the `gophercloud` library does not yet provide a way to do so. Conflicts with `local_file_path`.

- `min_disk_gb` - (Optional) Amount of disk space (in GB) required to boot image. Defaults to 0.
- `min_ram_mb` - (Optional) Amount of ram (in MB) required to boot image. Defauts to 0.
- `name` - (Required) The name of the image.
- `properties` - (Optional) A map of key/value pairs to set freeform information about an image.
- `protected` - (Optional) If true, image will not be deletable. Defaults to false.
- `region` - (Optional) The region in which to obtain the V2 Glance client. A Glance client is needed to create an Image that can be used with a compute instance. If omitted, the `region` argument of the provider is used. Changing this creates a new Image.
- `tags` - (Optional) The tags of the image. It must be a list of strings. At this time, it is not possible to delete all tags of an image.
- `visibility` - (Optional) The visibility of the image. Must be one of "public", "private", "community", or "shared". The ability to set the visibility depends upon the configuration of the OpenStack cloud.

» Attributes Reference

The following attributes are exported:

- `checksum` - The checksum of the data associated with the image.
- `container_format` - See Argument Reference above.
- `created_at` - The date the image was created.
- `disk_format` - See Argument Reference above.
- `file` - the trailing path after the glance endpoint that represent the location of the image or the path to retrieve it.
- `id` - A unique ID assigned by Glance.
- `metadata` - The metadata associated with the image. Image metadata allow for meaningfully define the image properties and tags. See http://docs.openstack.org/developer/glance/metadefs-concepts.html.
- `min_disk_gb` - See Argument Reference above.
- `min_ram_mb` - See Argument Reference above.
- `name` - See Argument Reference above.
- `owner` - The id of the openstack user who owns the image.
- `properties` - See Argument Reference above.
- `protected` - See Argument Reference above.
- `region` - See Argument Reference above.
- `schema` - The path to the JSON-schema that represent the image or image
- `size_bytes` - The size in bytes of the data associated with the image.

- `status` - The status of the image. It can be "queued", "active" or "saving".
- `tags` - See Argument Reference above.
- `update_at` - The date the image was last updated.
- `visibility` - See Argument Reference above.

» Import

Images can be imported using the `id`, e.g.

```
$ terraform import openstack_images_image_v2.rancheros 89c60255-9bd6-460c-822a-e2b959ede9d2
```

» openstack_networking_floatingip_v2

Manages a V2 floating IP resource within OpenStack Neutron (networking) that can be used for load balancers. These are similar to Nova (compute) floating IP resources, but only compute floating IPs can be used with compute instances.

» Example Usage

```
resource "openstack_networking_floatingip_v2" "floatip_1" {
  pool = "public"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a floating IP that can be used with another networking resource, such as a load balancer. If omitted, the `region` argument of the provider is used. Changing this creates a new floating IP (which may or may not have a different address).
- `pool` - (Required) The name of the pool from which to obtain the floating IP. Changing this creates a new floating IP.
- `port_id` - (Optional) ID of an existing port with at least one IP address to associate with this floating IP.
- `tenant_id` - (Optional) The target tenant ID in which to allocate the floating IP, if you specify this together with a port_id, make sure the target port belongs to the same tenant. Changing this creates a new floating IP (which may or may not have a different address)

- `fixed_ip` - Fixed IP of the port to associate with this floating IP. Required if the port has multiple fixed IPs.
- `subnet_id` - (Optional) The subnet ID of the floating IP pool. Specify this if the floating IP network has multiple subnets.
- `value_specs` - (Optional) Map of additional options.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `pool` - See Argument Reference above.
- `address` - The actual floating IP address itself.
- `port_id` - ID of associated port.
- `tenant_id` - the ID of the tenant in which to create the floating IP.
- `fixed_ip` - The fixed IP which the floating IP maps to.

» Import

Floating IPs can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_floatingip_v2.floatip_1 2c7f39f3-702b-48d1-940c-b503
```

» openstack_networking_network_v2

Manages a V2 Neutron network resource within OpenStack.

» Example Usage

```
resource "openstack_networking_network_v2" "network_1" {
  name           = "network_1"
  admin_state_up = "true"
}

resource "openstack_networking_subnet_v2" "subnet_1" {
  name       = "subnet_1"
  network_id = "${openstack_networking_network_v2.network_1.id}"
  cidr       = "192.168.199.0/24"
  ip_version = 4
}
```

```
resource "openstack_compute_secgroup_v2" "secgroup_1" {
  name        = "secgroup_1"
  description = "a security group"

  rule {
    from_port   = 22
    to_port     = 22
    ip_protocol = "tcp"
    cidr        = "0.0.0.0/0"
  }
}

resource "openstack_networking_port_v2" "port_1" {
  name               = "port_1"
  network_id         = "${openstack_networking_network_v2.network_1.id}"
  admin_state_up     = "true"
  security_group_ids = ["${openstack_compute_secgroup_v2.secgroup_1.id}"]

  fixed_ip {
    "subnet_id"  = "${openstack_networking_subnet_v2.subnet_1.id}"
    "ip_address" = "192.168.199.10"
  }
}

resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  security_groups = ["${openstack_compute_secgroup_v2.secgroup_1.name}"]

  network {
    port = "${openstack_networking_port_v2.port_1.id}"
  }
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a Neutron network. If omitted, the `region` argument of the provider is used. Changing this creates a new network.
- `name` - (Optional) The name of the network. Changing this updates the name of the existing network.
- `shared` - (Optional) Specifies whether the network resource can be ac-

cessed by any tenant or not. Changing this updates the sharing capabalities of the existing network.

- `tenant_id` - (Optional) The owner of the network. Required if admin wants to create a network for another tenant. Changing this creates a new network.
- `admin_state_up` - (Optional) The administrative state of the network. Acceptable values are "true" and "false". Changing this value updates the state of the existing network.
- `segments` - (Optional) An array of one or more provider segment objects.
- `value_specs` - (Optional) Map of additional options.
- `availability_zone_hints` - (Optional) An availability zone is used to make network resources highly available. Used for resources with high availability so that they are scheduled on different availability zones. Changing this creates a new network.

The `segments` block supports:

- `physical_network` - The phisical network where this network is implemented.
- `segmentation_id` - An isolated segment on the physical network.
- `network_type` - The type of physical network.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `shared` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `availability_zone_hints` - See Argument Reference above.

» Import

Networks can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_network_v2.network_1 d90ce693-5ccf-4136-a0ed-152ce41
```

» openstack_networking_port_v2

Manages a V2 port resource within OpenStack.

» Example Usage

```
resource "openstack_networking_network_v2" "network_1" {
  name           = "network_1"
  admin_state_up = "true"
}

resource "openstack_networking_port_v2" "port_1" {
  name           = "port_1"
  network_id     = "${openstack_networking_network_v2.network_1.id}"
  admin_state_up = "true"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 networking client. A networking client is needed to create a port. If omitted, the `region` argument of the provider is used. Changing this creates a new port.
- `name` - (Optional) A unique name for the port. Changing this updates the `name` of an existing port.
- `network_id` - (Required) The ID of the network to attach the port to. Changing this creates a new port.
- `admin_state_up` - (Optional) Administrative up/down status for the port (must be "true" or "false" if provided). Changing this updates the `admin_state_up` of an existing port.
- `mac_address` - (Optional) Specify a specific MAC address for the port. Changing this creates a new port.
- `tenant_id` - (Optional) The owner of the Port. Required if admin wants to create a port for another tenant. Changing this creates a new port.
- `device_owner` - (Optional) The device owner of the Port. Changing this creates a new port.
- `security_group_ids` - (Optional - Conflicts with `no_security_groups`) A list of security group IDs to apply to the port. The security groups must be specified by ID and not name (as opposed to how they are configured with the Compute Instance).
- `no_security_groups` - (Optional - Conflicts with `security_group_ids`) If set to `true`, then no security groups are applied to the port. If set to `false` and no `security_group_ids` are specified, then the Port will yield

to the default behavior of the Networking service, which is to usually apply the "default" security group.

- `device_id` - (Optional) The ID of the device attached to the port. Changing this creates a new port.
- `fixed_ip` - (Optional) An array of desired IPs for this port. The structure is described below.
- `allowed_address_pairs` - (Optional) An IP/MAC Address pair of additional IP addresses that can be active on this port. The structure is described below.
- `value_specs` - (Optional) Map of additional options.

The `fixed_ip` block supports:

- `subnet_id` - (Required) Subnet in which to allocate IP address for this port.
- `ip_address` - (Optional) IP address desired in the subnet for this port. If you don't specify `ip_address`, an available IP address from the specified subnet will be allocated to this port. This field will not be populated if it is left blank. To retrieve the assigned IP address, use the `all_fixed_ips` attribute.

The `allowed_address_pairs` block supports:

- `ip_address` - (Required) The additional IP address.
- `mac_address` - (Optional) The additional MAC address.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `mac_address` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `device_owner` - See Argument Reference above.
- `security_group_ids` - See Argument Reference above.
- `device_id` - See Argument Reference above.
- `fixed_ip` - See Argument Reference above.
- `all_fixed_ips` - The collection of Fixed IP addresses on the port in the order returned by the Network v2 API.
- `all_security_group_ids` - The collection of Security Group IDs on the port which have been explicitly and implicitly added.

» Import

Ports can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_port_v2.port_1 eae26a3e-1c33-4cc1-9c31-0cd729c438a1
```

» Notes

» Ports and Instances

There are some notes to consider when connecting Instances to networks using Ports. Please see the `openstack_compute_instance_v2` documentation for further documentation.

» openstack_networking_router_interface_v2

Manages a V2 router interface resource within OpenStack.

» Example Usage

```
resource "openstack_networking_network_v2" "network_1" {
  name           = "tf_test_network"
  admin_state_up = "true"
}

resource "openstack_networking_subnet_v2" "subnet_1" {
  network_id = "${openstack_networking_network_v2.network_1.id}"
  cidr       = "192.168.199.0/24"
  ip_version = 4
}

resource "openstack_networking_router_v2" "router_1" {
  name             = "my_router"
  external_gateway = "f67f0d72-0ddf-11e4-9d95-e1f29f417e2f"
}

resource "openstack_networking_router_interface_v2" "router_interface_1" {
  router_id = "${openstack_networking_router_v2.router_1.id}"
  subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 networking client. A networking client is needed to create a router. If omitted, the `region` argument of the provider is used. Changing this creates a new router interface.
- `router_id` - (Required) ID of the router this interface belongs to. Changing this creates a new router interface.
- `subnet_id` - ID of the subnet this interface connects to. Changing this creates a new router interface.
- `port_id` - ID of the port this interface connects to. Changing this creates a new router interface.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `router_id` - See Argument Reference above.
- `subnet_id` - See Argument Reference above.
- `port_id` - See Argument Reference above.

» Import

Router Interfaces can be imported using the port `id`, e.g.

```
$ openstack port list --router <router name or id>
$ terraform import openstack_networking_router_interface_v2.int_1 <port id from above output
```

» openstack_networking_router_route_v2

Creates a routing entry on a OpenStack V2 router.

» Example Usage

```
resource "openstack_networking_router_v2" "router_1" {
  name           = "router_1"
  admin_state_up = "true"
}
```

```
resource "openstack_networking_network_v2" "network_1" {
  name           = "network_1"
  admin_state_up = "true"
}

resource "openstack_networking_subnet_v2" "subnet_1" {
  network_id = "${openstack_networking_network_v2.network_1.id}"
  cidr       = "192.168.199.0/24"
  ip_version = 4
}

resource "openstack_networking_router_interface_v2" "int_1" {
  router_id = "${openstack_networking_router_v2.router_1.id}"
  subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}

resource "openstack_networking_router_route_v2" "router_route_1" {
  depends_on       = ["openstack_networking_router_interface_v2.int_1"]
  router_id        = "${openstack_networking_router_v2.router_1.id}"
  destination_cidr = "10.0.1.0/24"
  next_hop         = "192.168.199.254"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 networking client. A networking client is needed to configure a routing entry on a router. If omitted, the `region` argument of the provider is used. Changing this creates a new routing entry.
- `router_id` - (Required) ID of the router this routing entry belongs to. Changing this creates a new routing entry.
- `destination_cidr` - (Required) CIDR block to match on the packet's destination IP. Changing this creates a new routing entry.
- `next_hop` - (Required) IP address of the next hop gateway. Changing this creates a new routing entry.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `router_id` - See Argument Reference above.
- `destination_cidr` - See Argument Reference above.
- `next_hop` - See Argument Reference above.

» Notes

The `next_hop` IP address must be directly reachable from the router at the `openstack_networking_router_route_v2` resource creation time. You can ensure that by explicitly specifying a dependency on the `openstack_networking_router_interface_v2` resource that connects the next hop to the router, as in the example above.

» Import

Routing entries can be imported using a combined ID using the following format:
`<router_id>-route-<destination_cidr>-<next_hop>`

```
$ terraform import openstack_networking_router_route_v2.router_route_1 686fe248-386c-4f70-9f
```

» openstack_networking_router_v2

Manages a V2 router resource within OpenStack.

» Example Usage

```
resource "openstack_networking_router_v2" "router_1" {
  name                = "my_router"
  admin_state_up      = true
  external_network_id = "f67f0d72-0ddf-11e4-9d95-e1f29f417e2f"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 networking client. A networking client is needed to create a router. If omitted, the `region` argument of the provider is used. Changing this creates a new router.
- `name` - (Optional) A unique name for the router. Changing this updates the `name` of an existing router.

- `admin_state_up` - (Optional) Administrative up/down status for the router (must be "true" or "false" if provided). Changing this updates the `admin_state_up` of an existing router.
- `distributed` - (Optional) Indicates whether or not to create a distributed router. The default policy setting in Neutron restricts usage of this property to administrative users only.
- `external_gateway` - (Deprecated - use `external_network_id` instead) The network UUID of an external gateway for the router. A router with an external gateway is required if any compute instances or load balancers will be using floating IPs. Changing this updates the external gateway of an existing router.
- `external_network_id` - (Optional) The network UUID of an external gateway for the router. A router with an external gateway is required if any compute instances or load balancers will be using floating IPs. Changing this updates the external gateway of the router.
- `enable_snat` - (Optional) Enable Source NAT for the router. Valid values are "true" or "false". An `external_network_id` has to be set in order to set this property. Changing this updates the `enable_snat` of the router.
- `external_fixed_ip` - (Optional) An external fixed IP for the router. This can be repeated. The structure is described below. An `external_network_id` has to be set in order to set this property. Changing this updates the external fixed IPs of the router.
- `tenant_id` - (Optional) The owner of the floating IP. Required if admin wants to create a router for another tenant. Changing this creates a new router.
- `value_specs` - (Optional) Map of additional driver-specific options.
- `vendor_options` - (Optional) Map of additional vendor-specific options. Supported options are described below.
- `availability_zone_hints` - (Optional) An availability zone is used to make network resources highly available. Used for resources with high availability so that they are scheduled on different availability zones. Changing this creates a new router.

The `external_fixed_ip` block supports:

- `subnet_id` - (Optional) Subnet in which the fixed IP belongs to.
- `ip_address` - (Optional) The IP address to set on the router.

The `vendor_options` block supports:

- `set_router_gateway_after_create` - (Optional) Boolean to control whether the Router gateway is assigned during creation or updated after creation.

» Attributes Reference

The following attributes are exported:

- `id` - ID of the router.
- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `external_gateway` - See Argument Reference above.
- `external_network_id` - See Argument Reference above.
- `enable_snat` - See Argument Reference above.
- `external_fixed_ip` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `value_specs` - See Argument Reference above.
- `availability_zone_hints` - See Argument Reference above.

» Import

Routers can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_router_v2.router_1 014395cd-89fc-4c9b-96b7-13d1ee79c
```

» openstack_networking_subnet_v2

Manages a V2 Neutron subnet resource within OpenStack.

» Example Usage

```
resource "openstack_networking_network_v2" "network_1" {
  name           = "tf_test_network"
  admin_state_up = "true"
}

resource "openstack_networking_subnet_v2" "subnet_1" {
  network_id = "${openstack_networking_network_v2.network_1.id}"
  cidr       = "192.168.199.0/24"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a Neutron subnet. If omitted, the `region` argument of the provider is used. Changing this creates a new subnet.
- `network_id` - (Required) The UUID of the parent network. Changing this creates a new subnet.
- `cidr` - (Required) CIDR representing IP range for this subnet, based on IP version. Changing this creates a new subnet.
- `ip_version` - (Optional) IP version, either 4 (default) or 6. Changing this creates a new subnet.
- `ipv6_address_mode` - (Optional) The IPv6 address mode. Valid values are `dhcpv6-stateful`, `dhcpv6-stateless`, or `slaac`.
- `ipv6_ra_mode` - (Optional) The IPv6 Router Advertisement mode. Valid values are `dhcpv6-stateful`, `dhcpv6-stateless`, or `slaac`.
- `name` - (Optional) The name of the subnet. Changing this updates the name of the existing subnet.
- `tenant_id` - (Optional) The owner of the subnet. Required if admin wants to create a subnet for another tenant. Changing this creates a new subnet.
- `allocation_pools` - (Optional) An array of sub-ranges of CIDR available for dynamic allocation to ports. The allocation_pool object structure is documented below. Changing this creates a new subnet.
- `gateway_ip` - (Optional) Default gateway used by devices in this subnet. Leaving this blank and not setting `no_gateway` will cause a default gateway of `.1` to be used. Changing this updates the gateway IP of the existing subnet.
- `no_gateway` - (Optional) Do not set a gateway IP on this subnet. Changing this removes or adds a default gateway IP of the existing subnet.
- `enable_dhcp` - (Optional) The administrative state of the network. Acceptable values are "true" and "false". Changing this value enables or disables the DHCP capabilities of the existing subnet. Defaults to true.
- `dns_nameservers` - (Optional) An array of DNS name server names used by hosts in this subnet. Changing this updates the DNS name servers for the existing subnet.
- `host_routes` - (Optional) An array of routes that should be used by devices with IPs from this subnet (not including local subnet route). The host_route object structure is documented below. Changing this updates the host routes for the existing subnet.
- `subnetpool_id` - (Optional) The ID of the subnetpool associated with the subnet.

- `value_specs` - (Optional) Map of additional options.

The `allocation_pools` block supports:

- `start` - (Required) The starting address.
- `end` - (Required) The ending address.

The `host_routes` block supports:

- `destination_cidr` - (Required) The destination CIDR.
- `next_hop` - (Required) The next hop in the route.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `network_id` - See Argument Reference above.
- `cidr` - See Argument Reference above.
- `ip_version` - See Argument Reference above.
- `name` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `allocation_pools` - See Argument Reference above.
- `gateway_ip` - See Argument Reference above.
- `enable_dhcp` - See Argument Reference above.
- `dns_nameservers` - See Argument Reference above.
- `host_routes` - See Argument Reference above.
- `subnetpool_id` - See Argument Reference above.

» Import

Subnets can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_subnet_v2.subnet_1 da4faf16-5546-41e4-8330-4d0002b74
```

» openstack_networking_subnetpool_v2

Manages a V2 Neutron subnetpool resource within OpenStack.

» Example Usage

```
resource "openstack_networking_subnetpool_v2" "subnetpool_1" {
  name = "my_ipv6_pool"
```

```
  ip_version = 6
  prefixes = ["fdf7:b13d:dead:beef::/64", "fd65:86cc:a334:39b7::/64"]
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a Neutron subnetpool. If omitted, the `region` argument of the provider is used. Changing this creates a new subnetpool.
- `name` - (Required) The name of the subnetpool. Changing this updates the name of the existing subnetpool.
- `default_quota` - (Optional) The per-project quota on the prefix space that can be allocated from the subnetpool for project subnets. Changing this updates the default quota of the existing subnetpool.
- `project_id` - (Optional) The owner of the subnetpool. Required if admin wants to create a subnetpool for another project. Changing this creates a new subnetpool.
- `prefixes` - (Required) A list of subnet prefixes to assign to the subnetpool. Neutron API merges adjacent prefixes and treats them as a single prefix. Each subnet prefix must be unique among all subnet prefixes in all subnetpools that are associated with the address scope. Changing this updates the prefixes list of the existing subnetpool.
- `default_prefixlen` - (Optional) The size of the prefix to allocate when the cidr or prefixlen attributes are omitted when you create the subnet. Defaults to the MinPrefixLen. Changing this updates the default prefixlen of the existing subnetpool.
- `min_prefixlen` - (Optional) The smallest prefix that can be allocated from a subnetpool. For IPv4 subnetpools, default is 8. For IPv6 subnetpools, default is 64. Changing this updates the min prefixlen of the existing subnetpool.
- `max_prefixlen` - (Optional) The maximum prefix size that can be allocated from the subnetpool. For IPv4 subnetpools, default is 32. For IPv6 subnetpools, default is 128. Changing this updates the max prefixlen of the existing subnetpool.
- `address_scope_id` - (Optional) The Neutron address scope to assign to the subnetpool. Changing this updates the address scope id of the existing subnetpool.

- `shared` - (Optional) Indicates whether this subnetpool is shared across all projects. Changing this updates the shared status of the existing subnetpool.
- `description` - (Optional) The human-readable description for the subnetpool. Changing this updates the description of the existing subnetpool.
- `is_default` - (Optional) Indicates whether the subnetpool is default subnetpool or not. Changing this updates the default status of the existing subnetpool.
- `value_specs` - (Optional) Map of additional options.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `default_quota` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `created_at` - The time at which subnetpool was created.
- `updated_at` - The time at which subnetpool was created.
- `prefixes` - See Argument Reference above.
- `default_prefixlen` - See Argument Reference above.
- `min_prefixlen` - See Argument Reference above.
- `max_prefixlen` - See Argument Reference above.
- `address_scope_id` - See Argument Reference above.
- `ip_version` - The IP protocol version.
- `shared` - See Argument Reference above.
- `description` - See Argument Reference above.
- `is_default` - See Argument Reference above.
- `revision_number` - The revision number of the subnetpool.
- `value_specs` - See Argument Reference above.

» Import

Subnetpools can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_subnetpool_v2.subnetpool_1 832cb7f3-59fe-40cf-8f64-8
```

» openstack_networking_secgroup_v2

Manages a V2 neutron security group resource within OpenStack. Unlike Nova security groups, neutron separates the group from the rules and also allows an

admin to target a specific tenant_id.

» Example Usage

```
resource "openstack_networking_secgroup_v2" "secgroup_1" {
  name        = "secgroup_1"
  description = "My neutron security group"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 networking client. A networking client is needed to create a port. If omitted, the `region` argument of the provider is used. Changing this creates a new security group.
- `name` - (Required) A unique name for the security group.
- `description` - (Optional) A unique name for the security group.
- `tenant_id` - (Optional) The owner of the security group. Required if admin wants to create a port for another tenant. Changing this creates a new security group.
- `delete_default_rules` - (Optional) Whether or not to delete the default egress security rules. This is `false` by default. See the below note for more information.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.

» Default Security Group Rules

In most cases, OpenStack will create some egress security group rules for each new security group. These security group rules will not be managed by Terraform, so if you prefer to have *all* aspects of your infrastructure managed by

Terraform, set `delete_default_rules` to `true` and then create separate security group rules such as the following:

```
resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_v4" {
  direction = "egress"
  ethertype = "IPv4"
  security_group_id = "${openstack_networking_secgroup_v2.secgroup.id}"
}

resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_v6" {
  direction = "egress"
  ethertype = "IPv6"
  security_group_id = "${openstack_networking_secgroup_v2.secgroup.id}"
}
```

Please note that this behavior may differ depending on the configuration of the OpenStack cloud. The above illustrates the current default Neutron behavior. Some OpenStack clouds might provide additional rules and some might not provide any rules at all (in which case the `delete_default_rules` setting is moot).

» Import

Security Groups can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_secgroup_v2.secgroup_1 38809219-5e8a-4852-9139-6f461
```

» openstack_networking_secgroup_rule_v2

Manages a V2 neutron security group rule resource within OpenStack. Unlike Nova security groups, neutron separates the group from the rules and also allows an admin to target a specific tenant_id.

» Example Usage

```
resource "openstack_networking_secgroup_v2" "secgroup_1" {
  name        = "secgroup_1"
  description = "My neutron security group"
}

resource "openstack_networking_secgroup_rule_v2" "secgroup_rule_1" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
```

```
  port_range_min    = 22
  port_range_max    = 22
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = "${openstack_networking_secgroup_v2.secgroup_1.id}"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 networking client. A networking client is needed to create a port. If omitted, the `region` argument of the provider is used. Changing this creates a new security group rule.
- `direction` - (Required) The direction of the rule, valid values are **ingress** or **egress**. Changing this creates a new security group rule.
- `ethertype` - (Required) The layer 3 protocol type, valid values are **IPv4** or **IPv6**. Changing this creates a new security group rule.
- `protocol` - (Optional) The layer 4 protocol type, valid values are following. Changing this creates a new security group rule. This is required if you want to specify a port range.
  - **tcp**
  - **udp**
  - **icmp**
  - **ah**
  - **dccp**
  - **egp**
  - **esp**
  - **gre**
  - **igmp**
  - **ipv6-encap**
  - **ipv6-frag**
  - **ipv6-icmp**
  - **ipv6-nonxt**
  - **ipv6-opts**
  - **ipv6-route**
  - **ospf**
  - **pgm**
  - **rsvp**
  - **sctp**
  - **udplite**
  - **vrrp**

- `port_range_min` - (Optional) The lower part of the allowed port range, valid integer value needs to be between 1 and 65535. Changing this creates a new security group rule.
- `port_range_max` - (Optional) The higher part of the allowed port range, valid integer value needs to be between 1 and 65535. Changing this creates a new security group rule.
- `remote_ip_prefix` - (Optional) The remote CIDR, the value needs to be a valid CIDR (i.e. 192.168.0.0/16). Changing this creates a new security group rule.
- `remote_group_id` - (Optional) The remote group id, the value needs to be an Openstack ID of a security group in the same tenant. Changing this creates a new security group rule.
- `security_group_id` - (Required) The security group id the rule should belong to, the value needs to be an Openstack ID of a security group in the same tenant. Changing this creates a new security group rule.
- `tenant_id` - (Optional) The owner of the security group. Required if admin wants to create a port for another tenant. Changing this creates a new security group rule.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `direction` - See Argument Reference above.
- `ethertype` - See Argument Reference above.
- `protocol` - See Argument Reference above.
- `port_range_min` - See Argument Reference above.
- `port_range_max` - See Argument Reference above.
- `remote_ip_prefix` - See Argument Reference above.
- `remote_group_id` - See Argument Reference above.
- `security_group_id` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.

» Import

Security Group Rules can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_secgroup_rule_v2.secgroup_rule_1 aeb68ee3-6e9d-4256-
```

» openstack_lb_member_v1

Manages a V1 load balancer member resource within OpenStack.

» Example Usage

```
resource "openstack_lb_member_v1" "member_1" {
  pool_id = "d9415786-5f1a-428b-b35f-2f1523e146d2"
  address = "192.168.0.10"
  port    = 80
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an LB member. If omitted, the `region` argument of the provider is used. Changing this creates a new LB member.
- `pool_id` - (Required) The ID of the LB pool. Changing this creates a new member.
- `address` - (Required) The IP address of the member. Changing this creates a new member.
- `port` - (Required) An integer representing the port on which the member is hosted. Changing this creates a new member.
- `admin_state_up` - (Optional) The administrative state of the member. Acceptable values are 'true' and 'false'. Changing this value updates the state of the existing member.
- `tenant_id` - (Optional) The owner of the member. Required if admin wants to create a member for another tenant. Changing this creates a new member.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `pool_id` - See Argument Reference above.
- `address` - See Argument Reference above.
- `port` - See Argument Reference above.

- `admin_state_up` - See Argument Reference above.
- `weight` - The load balancing weight of the member. This is currently unable to be set through Terraform.

» Import

Load Balancer Members can be imported using the `id`, e.g.

```
$ terraform import openstack_lb_member_v1.member_1 a7498676-4fe4-4243-a864-2eaaf18c73df
```

» openstack_lb_monitor_v1

Manages a V1 load balancer monitor resource within OpenStack.

» Example Usage

```
resource "openstack_lb_monitor_v1" "monitor_1" {
  type           = "PING"
  delay          = 30
  timeout        = 5
  max_retries    = 3
  admin_state_up = "true"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an LB monitor. If omitted, the `region` argument of the provider is used. Changing this creates a new LB monitor.
- `type` - (Required) The type of probe, which is PING, TCP, HTTP, or HTTPS, that is sent by the monitor to verify the member state. Changing this creates a new monitor.
- `delay` - (Required) The time, in seconds, between sending probes to members. Changing this creates a new monitor.
- `timeout` - (Required) Maximum number of seconds for a monitor to wait for a ping reply before it times out. The value must be less than the delay value. Changing this updates the timeout of the existing monitor.

- `max_retries` - (Required) Number of permissible ping failures before changing the member's status to INACTIVE. Must be a number between 1 and 10. Changing this updates the max_retries of the existing monitor.
- `url_path` - (Optional) Required for HTTP(S) types. URI path that will be accessed if monitor type is HTTP or HTTPS. Changing this updates the url_path of the existing monitor.
- `http_method` - (Optional) Required for HTTP(S) types. The HTTP method used for requests by the monitor. If this attribute is not specified, it defaults to "GET". Changing this updates the http_method of the existing monitor.
- `expected_codes` - (Optional) equired for HTTP(S) types. Expected HTTP codes for a passing HTTP(S) monitor. You can either specify a single status like "200", or a range like "200-202". Changing this updates the expected_codes of the existing monitor.
- `admin_state_up` - (Optional) The administrative state of the monitor. Acceptable values are "true" and "false". Changing this value updates the state of the existing monitor.
- `tenant_id` - (Optional) The owner of the monitor. Required if admin wants to create a monitor for another tenant. Changing this creates a new monitor.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `type` - See Argument Reference above.
- `delay` - See Argument Reference above.
- `timeout` - See Argument Reference above.
- `max_retries` - See Argument Reference above.
- `url_path` - See Argument Reference above.
- `http_method` - See Argument Reference above.
- `expected_codes` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.

» Import

Load Balancer Members can be imported using the `id`, e.g.

```
$ terraform import openstack_lb_monitor_v1.monitor_1 119d7530-72e9-449a-aa97-124a5ef1992c
```

» openstack_lb_pool_v1

Manages a V1 load balancer pool resource within OpenStack.

» Example Usage

```
resource "openstack_lb_pool_v1" "pool_1" {
  name        = "tf_test_lb_pool"
  protocol    = "HTTP"
  subnet_id   = "12345"
  lb_method   = "ROUND_ROBIN"
  lb_provider = "haproxy"
  monitor_ids = ["67890"]
}
```

» Complete Load Balancing Stack Example

```
resource "openstack_networking_network_v2" "network_1" {
  name           = "network_1"
  admin_state_up = "true"
}

resource "openstack_networking_subnet_v2" "subnet_1" {
  network_id = "${openstack_networking_network_v2.network_1.id}"
  cidr       = "192.168.199.0/24"
  ip_version = 4
}

resource "openstack_compute_secgroup_v2" "secgroup_1" {
  name        = "secgroup_1"
  description = "Rules for secgroup_1"

  rule {
    from_port   = -1
    to_port     = -1
    ip_protocol = "icmp"
    cidr        = "0.0.0.0/0"
  }

  rule {
    from_port   = 80
    to_port     = 80
    ip_protocol = "tcp"
    cidr        = "0.0.0.0/0"
```

```
  }
}

resource "openstack_compute_instance_v2" "instance_1" {
  name            = "instance_1"
  security_groups = ["default", "${openstack_compute_secgroup_v2.secgroup_1.name}"]

  network {
    uuid = "${openstack_networking_network_v2.network_1.id}"
  }
}

resource "openstack_compute_instance_v2" "instance_2" {
  name            = "instance_2"
  security_groups = ["default", "${openstack_compute_secgroup_v2.secgroup_1.name}"]

  network {
    uuid = "${openstack_networking_network_v2.network_1.id}"
  }
}

resource "openstack_lb_monitor_v1" "monitor_1" {
  type           = "TCP"
  delay          = 30
  timeout        = 5
  max_retries    = 3
  admin_state_up = "true"
}

resource "openstack_lb_pool_v1" "pool_1" {
  name        = "pool_1"
  protocol    = "TCP"
  subnet_id   = "${openstack_networking_subnet_v2.subnet_1.id}"
  lb_method   = "ROUND_ROBIN"
  monitor_ids = ["${openstack_lb_monitor_v1.monitor_1.id}"]
}

resource "openstack_lb_member_v1" "member_1" {
  pool_id = "${openstack_lb_pool_v1.pool_1.id}"
  address = "${openstack_compute_instance_v2.instance_1.access_ip_v4}"
  port    = 80
}

resource "openstack_lb_member_v1" "member_2" {
  pool_id = "${openstack_lb_pool_v1.pool_1.id}"
  address = "${openstack_compute_instance_v2.instance_2.access_ip_v4}"
```

```
  port    = 80
}

resource "openstack_lb_vip_v1" "vip_1" {
  name      = "vip_1"
  subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
  protocol  = "TCP"
  port      = 80
  pool_id   = "${openstack_lb_pool_v1.pool_1.id}"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an LB pool. If omitted, the `region` argument of the provider is used. Changing this creates a new LB pool.
- `name` - (Required) The name of the pool. Changing this updates the name of the existing pool.
- `protocol` - (Required) The protocol used by the pool members, you can use either 'TCP, 'HTTP', or 'HTTPS'. Changing this creates a new pool.
- `subnet_id` - (Required) The network on which the members of the pool will be located. Only members that are on this network can be added to the pool. Changing this creates a new pool.
- `lb_method` - (Required) The algorithm used to distribute load between the members of the pool. The current specification supports 'ROUND_ROBIN' and 'LEAST_CONNECTIONS' as valid values for this attribute.
- `lb_provider` - (Optional) The backend load balancing provider. For example: `haproxy`, `F5`, etc.
- `tenant_id` - (Optional) The owner of the pool. Required if admin wants to create a pool member for another tenant. Changing this creates a new pool.
- `monitor_ids` - (Optional) A list of IDs of monitors to associate with the pool.
- `member` - (Optional) An existing node to add to the pool. Changing this updates the members of the pool. The member object structure is documented below. Please note that the `member` block is deprecated in favor of the `openstack_lb_member_v1` resource.

The `member` block supports:

- `address` - (Required) The IP address of the member. Changing this creates a new member.
- `port` - (Required) An integer representing the port on which the member is hosted. Changing this creates a new member.
- `admin_state_up` - (Required) The administrative state of the member. Acceptable values are 'true' and 'false'. Changing this value updates the state of the existing member.
- `tenant_id` - (Optional) The owner of the member. Required if admin wants to create a pool member for another tenant. Changing this creates a new member.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `protocol` - See Argument Reference above.
- `subnet_id` - See Argument Reference above.
- `lb_method` - See Argument Reference above.
- `lb_provider` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `monitor_id` - See Argument Reference above.
- `member` - See Argument Reference above.

» Notes

The `member` block is deprecated in favor of the `openstack_lb_member_v1` resource.

» Import

Load Balancer Pools can be imported using the `id`, e.g.

```
$ terraform import openstack_lb_pool_v1.pool_1 b255e6ba-02ad-43e6-8951-3428ca26b713
```

» openstack_lb_vip_v1

Manages a V1 load balancer vip resource within OpenStack.

» Example Usage

```
resource "openstack_lb_vip_v1" "vip_1" {
  name      = "tf_test_lb_vip"
  subnet_id = "12345"
  protocol  = "HTTP"
  port      = 80
  pool_id   = "67890"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a VIP. If omitted, the `region` argument of the provider is used. Changing this creates a new VIP.
- `name` - (Required) The name of the vip. Changing this updates the name of the existing vip.
- `subnet_id` - (Required) The network on which to allocate the vip's address. A tenant can only create vips on networks authorized by policy (e.g. networks that belong to them or networks that are shared). Changing this creates a new vip.
- `protocol` - (Required) The protocol - can be either 'TCP, 'HTTP', or HTTPS'. Changing this creates a new vip.
- `port` - (Required) The port on which to listen for client traffic. Changing this creates a new vip.
- `pool_id` - (Required) The ID of the pool with which the vip is associated. Changing this updates the pool_id of the existing vip.
- `tenant_id` - (Optional) The owner of the vip. Required if admin wants to create a vip member for another tenant. Changing this creates a new vip.
- `address` - (Optional) The IP address of the vip. Changing this creates a new vip.
- `description` - (Optional) Human-readable description for the vip. Changing this updates the description of the existing vip.
- `persistence` - (Optional) Omit this field to prevent session persistence. The persistence object structure is documented below. Changing this updates the persistence of the existing vip.

- `conn_limit` - (Optional) The maximum number of connections allowed for the vip. Default is -1, meaning no limit. Changing this updates the conn_limit of the existing vip.
- `floating_ip` - (Optional) A *Networking* Floating IP that will be associated with the vip. The Floating IP must be provisioned already.
- `admin_state_up` - (Optional) The administrative state of the vip. Acceptable values are "true" and "false". Changing this value updates the state of the existing vip.

The `persistence` block supports:

- `type` - (Required) The type of persistence mode. Valid values are "SOURCE_IP", "HTTP_COOKIE", or "APP_COOKIE".
- `cookie_name` - (Optional) The name of the cookie if persistence mode is set appropriately.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `subnet_id` - See Argument Reference above.
- `protocol` - See Argument Reference above.
- `port` - See Argument Reference above.
- `pool_id` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `address` - See Argument Reference above.
- `description` - See Argument Reference above.
- `persistence` - See Argument Reference above.
- `conn_limit` - See Argument Reference above.
- `floating_ip` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `port_id` - Port UUID for this VIP at associated floating IP (if any).

» Import

Load Balancer VIPs can be imported using the `id`, e.g.

```
$ terraform import openstack_lb_vip_v1.vip_1 50e16b26-89c1-475e-a492-76167182511e
```

» openstack_lb_loadbalancer_v2

Manages a V2 loadbalancer resource within OpenStack.

» Example Usage

```
resource "openstack_lb_loadbalancer_v2" "lb_1" {
  vip_subnet_id = "d9415786-5f1a-428b-b35f-2f1523e146d2"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an LB member. If omitted, the `region` argument of the provider is used. Changing this creates a new LB member.
- `vip_subnet_id` - (Required) The network on which to allocate the Loadbalancer's address. A tenant can only create Loadbalancers on networks authorized by policy (e.g. networks that belong to them or networks that are shared). Changing this creates a new loadbalancer.
- `name` - (Optional) Human-readable name for the Loadbalancer. Does not have to be unique.
- `description` - (Optional) Human-readable description for the Loadbalancer.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the Loadbalancer. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new loadbalancer.
- `vip_address` - (Optional) The ip address of the load balancer. Changing this creates a new loadbalancer.
- `admin_state_up` - (Optional) The administrative state of the Loadbalancer. A valid value is true (UP) or false (DOWN).
- `flavor` - (Optional) The UUID of a flavor. Changing this creates a new loadbalancer.
- `loadbalancer_provider` - (Optional) The name of the provider. Changing this creates a new loadbalancer.
- `security_group_ids` - (Optional) A list of security group IDs to apply to the loadbalancer. The security groups must be specified by ID and not name (as opposed to how they are configured with the Compute Instance).

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `vip_subnet_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `vip_address` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `flavor` - See Argument Reference above.
- `loadbalancer_provider` - See Argument Reference above.
- `security_group_ids` - See Argument Reference above.
- `vip_port_id` - The Port ID of the Load Balancer IP.

» openstack_lb_listener_v2

Manages a V2 listener resource within OpenStack.

» Example Usage

```
resource "openstack_lb_listener_v2" "listener_1" {
  protocol        = "HTTP"
  protocol_port   = 8080
  loadbalancer_id = "d9415786-5f1a-428b-b35f-2f1523e146d2"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an . If omitted, the `region` argument of the provider is used. Changing this creates a new Listener.
- `protocol` - (Required) The protocol - can either be TCP, HTTP, HTTPS or TERMINATED_HTTPS. Changing this creates a new Listener.
- `protocol_port` - (Required) The port on which to listen for client traffic. Changing this creates a new Listener.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the Listener. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new Listener.

- `loadbalancer_id` - (Required) The load balancer on which to provision this Listener. Changing this creates a new Listener.
- `name` - (Optional) Human-readable name for the Listener. Does not have to be unique.
- `default_pool_id` - (Optional) The ID of the default pool with which the Listener is associated. Changing this creates a new Listener.
- `description` - (Optional) Human-readable description for the Listener.
- `connection_limit` - (Optional) The maximum number of connections allowed for the Listener.
- `default_tls_container_ref` - (Optional) A reference to a Barbican Secrets container which stores TLS information. This is required if the protocol is `TERMINATED_HTTPS`. See here for more information.
- `sni_container_refs` - (Optional) A list of references to Barbican Secrets containers which store SNI information. See here for more information.
- `admin_state_up` - (Optional) The administrative state of the Listener. A valid value is true (UP) or false (DOWN).

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID for the Listener.
- `protocol` - See Argument Reference above.
- `protocol_port` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `default_port_id` - See Argument Reference above.
- `description` - See Argument Reference above.
- `connection_limit` - See Argument Reference above.
- `default_tls_container_ref` - See Argument Reference above.
- `sni_container_refs` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.

» openstack_lb_pool_v2

Manages a V2 pool resource within OpenStack.

» Example Usage

```
resource "openstack_lb_pool_v2" "pool_1" {
  protocol    = "HTTP"
  lb_method   = "ROUND_ROBIN"
  listener_id = "d9415786-5f1a-428b-b35f-2f1523e146d2"

  persistence {
    type        = "HTTP_COOKIE"
    cookie_name = "testCookie"
  }
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an . If omitted, the `region` argument of the provider is used. Changing this creates a new pool.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the pool. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new pool.
- `name` - (Optional) Human-readable name for the pool.
- `description` - (Optional) Human-readable description for the pool.
- `protocol` = (Required) The protocol - can either be TCP, HTTP or HTTPS. Changing this creates a new pool.
- `loadbalancer_id` - (Optional) The load balancer on which to provision this pool. Changing this creates a new pool. Note: One of LoadbalancerID or ListenerID must be provided.
- `listener_id` - (Optional) The Listener on which the members of the pool will be associated with. Changing this creates a new pool. Note: One of LoadbalancerID or ListenerID must be provided.
- `lb_method` - (Required) The load balancing algorithm to distribute traffic to the pool's members. Must be one of ROUND_ROBIN, LEAST_CONNECTIONS, or SOURCE_IP.
- `persistence` - Omit this field to prevent session persistence. Indicates whether connections in the same session will be processed by the same Pool member or not. Changing this creates a new pool.
- `admin_state_up` - (Optional) The administrative state of the pool. A valid value is true (UP) or false (DOWN).

The `persistence` argument supports:

- `type` - (Required) The type of persistence mode. The current specification supports SOURCE_IP, HTTP_COOKIE, and APP_COOKIE.
- `cookie_name` - (Optional) The name of the cookie if persistence mode is set appropriately. Required if `type = APP_COOKIE`.

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID for the pool.
- `tenant_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `protocol` - See Argument Reference above.
- `lb_method` - See Argument Reference above.
- `persistence` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.

» openstack_lb_member_v2

Manages a V2 member resource within OpenStack.

» Example Usage

```
resource "openstack_lb_member_v2" "member_1" {
  address       = "192.168.199.23"
  protocol_port = 8080
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an . If omitted, the `region` argument of the provider is used. Changing this creates a new member.
- `pool_id` - (Required) The id of the pool that this member will be assigned to.
- `subnet_id` - (Optional) The subnet in which to access the member

- `name` - (Optional) Human-readable name for the member.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the member. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new member.
- `address` - (Required) The IP address of the member to receive traffic from the load balancer. Changing this creates a new member.
- `protocol_port` - (Required) The port on which to listen for client traffic. Changing this creates a new member.
- `weight` - (Optional) A positive integer value that indicates the relative portion of traffic that this member should receive from the pool. For example, a member with a weight of 10 receives five times as much traffic as a member with a weight of 2.
- `admin_state_up` - (Optional) The administrative state of the member. A valid value is true (UP) or false (DOWN).

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID for the member.
- `name` - See Argument Reference above.
- `weight` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `subnet_id` - See Argument Reference above.
- `pool_id` - See Argument Reference above.
- `address` - See Argument Reference above.
- `protocol_port` - See Argument Reference above.

» openstack_lb_monitor_v2

Manages a V2 monitor resource within OpenStack.

» Example Usage

```
resource "openstack_lb_monitor_v2" "monitor_1" {
  pool_id     = "${openstack_lb_pool_v2.pool_1.id}"
  type        = "PING"
  delay       = 20
  timeout     = 10
```

```
  max_retries = 5
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create an . If omitted, the `region` argument of the provider is used. Changing this creates a new monitor.
- `pool_id` - (Required) The id of the pool that this monitor will be assigned to.
- `name` - (Optional) The Name of the Monitor.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the monitor. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new monitor.
- `type` - (Required) The type of probe, which is PING, TCP, HTTP, or HTTPS, that is sent by the load balancer to verify the member state. Changing this creates a new monitor.
- `delay` - (Required) The time, in seconds, between sending probes to members.
- `timeout` - (Required) Maximum number of seconds for a monitor to wait for a ping reply before it times out. The value must be less than the delay value.
- `max_retries` - (Required) Number of permissible ping failures before changing the member's status to INACTIVE. Must be a number between 1 and 10..
- `url_path` - (Optional) Required for HTTP(S) types. URI path that will be accessed if monitor type is HTTP or HTTPS.
- `http_method` - (Optional) Required for HTTP(S) types. The HTTP method used for requests by the monitor. If this attribute is not specified, it defaults to "GET".
- `expected_codes` - (Optional) Required for HTTP(S) types. Expected HTTP codes for a passing HTTP(S) monitor. You can either specify a single status like "200", or a range like "200-202".
- `admin_state_up` - (Optional) The administrative state of the monitor. A valid value is true (UP) or false (DOWN).

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID for the monitor.
- `tenant_id` - See Argument Reference above.
- `type` - See Argument Reference above.
- `delay` - See Argument Reference above.
- `timeout` - See Argument Reference above.
- `max_retries` - See Argument Reference above.
- `url_path` - See Argument Reference above.
- `http_method` - See Argument Reference above.
- `expected_codes` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.

» openstack_fw_firewall_v1

Manages a v1 firewall resource within OpenStack.

» Example Usage

```
resource "openstack_fw_rule_v1" "rule_1" {
  name             = "my-rule-1"
  description      = "drop TELNET traffic"
  action           = "deny"
  protocol         = "tcp"
  destination_port = "23"
  enabled          = "true"
}

resource "openstack_fw_rule_v1" "rule_2" {
  name             = "my-rule-2"
  description      = "drop NTP traffic"
  action           = "deny"
  protocol         = "udp"
  destination_port = "123"
  enabled          = "false"
}

resource "openstack_fw_policy_v1" "policy_1" {
  name = "my-policy"

  rules = ["${openstack_fw_rule_v1.rule_1.id}",
    "${openstack_fw_rule_v1.rule_2.id}",
```

```
  ]
}

resource "openstack_fw_firewall_v1" "firewall_1" {
  name      = "my-firewall"
  policy_id = "${openstack_fw_policy_v1.policy_1.id}"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the v1 networking client. A networking client is needed to create a firewall. If omitted, the `region` argument of the provider is used. Changing this creates a new firewall.
- `policy_id` - (Required) The policy resource id for the firewall. Changing this updates the `policy_id` of an existing firewall.
- `name` - (Optional) A name for the firewall. Changing this updates the `name` of an existing firewall.
- `description` - (Required) A description for the firewall. Changing this updates the `description` of an existing firewall.
- `admin_state_up` - (Optional) Administrative up/down status for the firewall (must be "true" or "false" if provided - defaults to "true"). Changing this updates the `admin_state_up` of an existing firewall.
- `tenant_id` - (Optional) The owner of the floating IP. Required if admin wants to create a firewall for another tenant. Changing this creates a new firewall.
- `associated_routers` - (Optional) Router(s) to associate this firewall instance with. Must be a list of strings. Changing this updates the associated routers of an existing firewall. Conflicts with `no_routers`.
- `no_routers` - (Optional) Should this firewall not be associated with any routers (must be "true" or "false" if provide - defaults to "false"). Conflicts with `associated_routers`.
- `value_specs` - (Optional) Map of additional options.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `policy_id` - See Argument Reference above.

- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `associated_routers` - See Argument Reference above.
- `no_routers` - See Argument Reference above.

» Import

Firewalls can be imported using the `id`, e.g.

```
$ terraform import openstack_fw_firewall_v1.firewall_1 c9e39fb2-ce20-46c8-a964-25f3898c7a97
```

» openstack_fw_policy_v1

Manages a v1 firewall policy resource within OpenStack.

» Example Usage

```
resource "openstack_fw_rule_v1" "rule_1" {
  name             = "my-rule-1"
  description      = "drop TELNET traffic"
  action           = "deny"
  protocol         = "tcp"
  destination_port = "23"
  enabled          = "true"
}

resource "openstack_fw_rule_v1" "rule_2" {
  name             = "my-rule-2"
  description      = "drop NTP traffic"
  action           = "deny"
  protocol         = "udp"
  destination_port = "123"
  enabled          = "false"
}

resource "openstack_fw_policy_v1" "policy_1" {
  name = "my-policy"

  rules = ["${openstack_fw_rule_v1.rule_1.id}",
    "${openstack_fw_rule_v1.rule_2.id}",
  ]
```

```
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the v1 networking client. A networking client is needed to create a firewall policy. If omitted, the `region` argument of the provider is used. Changing this creates a new firewall policy.
- `name` - (Optional) A name for the firewall policy. Changing this updates the `name` of an existing firewall policy.
- `description` - (Optional) A description for the firewall policy. Changing this updates the `description` of an existing firewall policy.
- `rules` - (Optional) An array of one or more firewall rules that comprise the policy. Changing this results in adding/removing rules from the existing firewall policy.
- `audited` - (Optional) Audit status of the firewall policy (must be "true" or "false" if provided - defaults to "false"). This status is set to "false" whenever the firewall policy or any of its rules are changed. Changing this updates the `audited` status of an existing firewall policy.
- `shared` - (Optional) Sharing status of the firewall policy (must be "true" or "false" if provided). If this is "true" the policy is visible to, and can be used in, firewalls in other tenants. Changing this updates the `shared` status of an existing firewall policy. Only administrative users can specify if the policy should be shared.
- `value_specs` - (Optional) Map of additional options.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `audited` - See Argument Reference above.
- `shared` - See Argument Reference above.

» Import

Firewall Policies can be imported using the `id`, e.g.

```
$ terraform import openstack_fw_policy_v1.policy_1 07f422e6-c596-474b-8b94-fe2c12506ce0
```

» openstack_fw_rule_v1

Manages a v1 firewall rule resource within OpenStack.

» Example Usage

```
resource "openstack_fw_rule_v1" "rule_1" {
  name             = "my_rule"
  description      = "drop TELNET traffic"
  action           = "deny"
  protocol         = "tcp"
  destination_port = "23"
  enabled          = "true"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the v1 Compute client. A Compute client is needed to create a firewall rule. If omitted, the `region` argument of the provider is used. Changing this creates a new firewall rule.
- `name` - (Optional) A unique name for the firewall rule. Changing this updates the `name` of an existing firewall rule.
- `description` - (Optional) A description for the firewall rule. Changing this updates the `description` of an existing firewall rule.
- `protocol` - (Required) The protocol type on which the firewall rule operates. Valid values are: `tcp`, `udp`, `icmp`, and `any`. Changing this updates the `protocol` of an existing firewall rule.
- `action` - (Required) Action to be taken (must be ”allow” or ”deny”) when the firewall rule matches. Changing this updates the `action` of an existing firewall rule.
- `ip_version` - (Optional) IP version, either 4 (default) or 6. Changing this updates the `ip_version` of an existing firewall rule.
- `source_ip_address` - (Optional) The source IP address on which the firewall rule operates. Changing this updates the `source_ip_address` of an existing firewall rule.

- `destination_ip_address` - (Optional) The destination IP address on which the firewall rule operates. Changing this updates the `destination_ip_address` of an existing firewall rule.
- `source_port` - (Optional) The source port on which the firewall rule operates. Changing this updates the `source_port` of an existing firewall rule.
- `destination_port` - (Optional) The destination port on which the firewall rule operates. Changing this updates the `destination_port` of an existing firewall rule.
- `enabled` - (Optional) Enabled status for the firewall rule (must be "true" or "false" if provided - defaults to "true"). Changing this updates the `enabled` status of an existing firewall rule.
- `tenant_id` - (Optional) The owner of the firewall rule. Required if admin wants to create a firewall rule for another tenant. Changing this creates a new firewall rule.
- `value_specs` - (Optional) Map of additional options.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `protocol` - See Argument Reference above.
- `action` - See Argument Reference above.
- `ip_version` - See Argument Reference above.
- `source_ip_address` - See Argument Reference above.
- `destination_ip_address` - See Argument Reference above.
- `source_port` - See Argument Reference above.
- `destination_port` - See Argument Reference above.
- `enabled` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.

» Import

Firewall Rules can be imported using the `id`, e.g.

```
$ terraform import openstack_fw_rule_v1.rule_1 8dbc0c28-e49c-463f-b712-5c5d1bbac327
```

» openstack_objectstorage_container_v1

Manages a V1 container resource within OpenStack.

» Example Usage

```
resource "openstack_objectstorage_container_v1" "container_1" {
  region = "RegionOne"
  name   = "tf-test-container-1"

  metadata {
    test = "true"
  }

  content_type = "application/json"
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to create the container. If omitted, the `region` argument of the provider is used. Changing this creates a new container.
- `name` - (Required) A unique name for the container. Changing this creates a new container.
- `container_read` - (Optional) Sets an access control list (ACL) that grants read access. This header can contain a comma-delimited list of users that can read the container (allows the GET method for all objects in the container). Changing this updates the access control list read access.
- `container_sync_to` - (Optional) The destination for container synchronization. Changing this updates container synchronization.
- `container_sync_key` - (Optional) The secret key for container synchronization. Changing this updates container synchronization.
- `container_write` - (Optional) Sets an ACL that grants write access. Changing this updates the access control list write access.
- `metadata` - (Optional) Custom key/value pairs to associate with the container. Changing this updates the existing container metadata.
- `content_type` - (Optional) The MIME type for the container. Changing this updates the MIME type.

- `force_destroy` - (Optional, Default:false) A boolean that indicates all objects should be deleted from the container so that the container can be destroyed without error. These objects are not recoverable.

» Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `container_read` - See Argument Reference above.
- `container_sync_to` - See Argument Reference above.
- `container_sync_key` - See Argument Reference above.
- `container_write` - See Argument Reference above.
- `metadata` - See Argument Reference above.
- `content_type` - See Argument Reference above.

» openstack_objectstorage_object_v1

Manages a V1 container object resource within OpenStack.

» Example Usage

» Example with simple content

```
resource "openstack_objectstorage_container_v1" "container_1" {
  region = "RegionOne"
  name   = "tf-test-container-1"

  metadata {
    test = "true"
  }

  content_type = "application/json"
}

resource "openstack_objectstorage_object_v1" "doc_1" {
  region         = "RegionOne"
  container_name = "${openstack_objectstorage_container_v1.container_1.name}"
  name           = "test/default.json"
  metadata {
    test = "true"
  }
```

```
  content_type = "application/json"
  content = <<JSON
              {
                "foo" : "bar"
              }
JSON

}
```

» Example with content from file

```
resource "openstack_objectstorage_container_v1" "container_1" {
  region = "RegionOne"
  name   = "tf-test-container-1"

  metadata {
    test = "true"
  }

  content_type = "application/json"
}

resource "openstack_objectstorage_object_v1" "doc_1" {
  region         = "RegionOne"
  container_name = "${openstack_objectstorage_container_v1.container_1.name}"
  name           = "test/default.json"
  metadata {
    test = "true"
  }

  content_type = "application/json"
  source       = "./default.json"
}
```

» Argument Reference

The following arguments are supported:

- `container_name` - (Required) A unique (within an account) name for the container. The container name must be from 1 to 256 characters long and can start with any character and contain any pattern. Character set must be UTF-8. The container name cannot contain a slash (/) character because this character delimits the container and object name. For example,

the path /v1/account/www/pages specifies the www container, not the www/pages container.

- `content` - (Optional) A string representing the content of the object. Conflicts with `source` and `copy_from`.
- `content_disposition` - (Optional) A string which specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default.
- `content_encoding` - (Optional) A string representing the value of the Content-Encoding metadata.
- `content_type` - (Optional) A string which sets the MIME type for the object.
- `copy_from` - (Optional) A string representing the name of an object used to create the new object by copying the `copy_from` object. The value is in form {container}/{object}. You must UTF-8-encode and then URL-encode the names of the container and object before you include them in the header. Conflicts with `source` and `content`.
- `delete_after` - (Optional) An integer representing the number of seconds after which the system removes the object. Internally, the Object Storage system stores this value in the X-Delete-At metadata item.
- `delete_at` - (Optional) An string representing the date when the system removes the object. For example, "2015-08-26" is equivalent to Mon, Wed, 26 Aug 2015 00:00:00 GMT.
- `detect_content_type` - (Optional) If set to true, Object Storage guesses the content type based on the file extension and ignores the value sent in the Content-Type header, if present.
- `etag` - (Optional) Used to trigger updates. The only meaningful value is ${md5(file("path/to/file"))}.
- `name` - (Required) A unique name for the object.
- `object_manifest` - (Optional) A string set to specify that this is a dynamic large object manifest object. The value is the container and object name prefix of the segment objects in the form container/prefix. You must UTF-8-encode and then URL-encode the names of the container and prefix before you include them in this header.
- `region` - (Optional) The region in which to create the container. If omitted, the `region` argument of the provider is used. Changing this creates a new container.
- `source` - (Optional) A string representing the local path of a file which will be used as the object's content. Conflicts with `source` and `copy_from`.

» Attributes Reference

The following attributes are exported:

- `content_length` - If the operation succeeds, this value is zero (0) or the length of informational or error text in the response body.
- `content_type` - If the operation succeeds, this value is the MIME type of the object. If the operation fails, this value is the MIME type of the error text in the response body.
- `date` - The date and time the system responded to the request, using the preferred format of RFC 7231 as shown in this example Thu, 16 Jun 2016 15:10:38 GMT. The time is always in UTC.
- `etag` - Whatever the value given in argument, will be overriden by the MD5 checksum of the uploaded object content. The value is not quoted. If it is an SLO, it would be MD5 checksum of the segments' etags.
- `last_modified` - The date and time when the object was last modified. The date and time stamp format is ISO 8601: CCYY-MM-DDThh:mm:ss±hh:mm For example, 2015-08-27T09:49:58-05:00. The ±hh:mm value, if included, is the time zone as an offset from UTC. In the previous example, the offset value is -05:00.
- `static_large_object` - True if object is a multipart_manifest.
- `trans_id` - A unique transaction ID for this request. Your service provider might need this value if you report a problem.
- `container_name` - See Argument Reference above.
- `content` - See Argument Reference above.
- `content_disposition` - See Argument Reference above.
- `content_encoding` - See Argument Reference above.
- `copy_from` - See Argument Reference above.
- `delete_after` - See Argument Reference above.
- `delete_at` - See Argument Reference above.
- `detect_content_type` - See Argument Reference above.
- `name` - See Argument Reference above.
- `object_manifest` - See Argument Reference above.
- `region` - See Argument Reference above.
- `source` - See Argument Reference above.