OpenStack Provider

The OpenStack provider is used to interact with the many resources supported by OpenStack. The provider needs to be configured with the proper credentials before it can be used.

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

Example Usage

```
# Configure the OpenStack Provider
provider "openstack" {
  user_name   = "admin"
  tenant_name = "admin"
  password    = "pwd"
  auth_url    = "http://myauthurl:5000/v2.0"
  region      = "RegionOne"
}

# Create a web server
resource "openstack_compute_instance_v2" "test-server" {
  # ...
}
```

Configuration Reference

The following arguments are supported:

- `auth_url` - (Optional; required if `cloud` is not specified) The Identity authentication URL. If omitted, the `OS_AUTH_URL` environment variable is used.
- `cloud` - (Optional; required if `auth_url` is not specified) An entry in a `clouds.yaml` file. See the OpenStack `os-client-config` documentation (https://docs.openstack.org/os-client-config/latest/user/configuration.html) for more information about `clouds.yaml` files. If omitted, the `OS_CLOUD` environment variable is used.
- `region` - (Optional) The region of the OpenStack cloud to use. If omitted, the `OS_REGION_NAME` environment variable is used. If `OS_REGION_NAME` is not set, then no region will be used. It should be possible to omit the region in single-region OpenStack environments, but this behavior may vary depending on the OpenStack environment being used.
- `user_name` - (Optional) The Username to login with. If omitted, the `OS_USERNAME` environment variable is used.
- `user_id` - (Optional) The User ID to login with. If omitted, the `OS_USER_ID` environment variable is used.
- `application_credential_id` - (Optional) (Identity v3 only) The ID of an application credential to authenticate with. An `application_credential_secret` has to bet set along with this parameter.
- `application_credential_name` - (Optional) (Identity v3 only) The name of an application credential to authenticate with. Conflicts with the `application_credential_name`, requires `user_id`, or `user_name` and `user_domain_name` (or `user_domain_id`) to be set.

- `application_credential_secret` - (Optional) (Identity v3 only) The secret of an application credential to authenticate with. Required by `application_credential_id` or `application_credential_name` .
- `tenant_id` - (Optional) The ID of the Tenant (Identity v2) or Project (Identity v3) to login with. If omitted, the `OS_TENANT_ID` or `OS_PROJECT_ID` environment variables are used.
- `tenant_name` - (Optional) The Name of the Tenant (Identity v2) or Project (Identity v3) to login with. If omitted, the `OS_TENANT_NAME` or `OS_PROJECT_NAME` environment variable are used.
- `password` - (Optional) The Password to login with. If omitted, the `OS_PASSWORD` environment variable is used.
- `token` - (Optional; Required if not using `user_name` and `password`) A token is an expiring, temporary means of access issued via the Keystone service. By specifying a token, you do not have to specify a username/password combination, since the token was already created by a username/password out of band of Terraform. If omitted, the `OS_TOKEN` or `OS_AUTH_TOKEN` environment variables are used.
- `user_domain_name` - (Optional) The domain name where the user is located. If omitted, the `OS_USER_DOMAIN_NAME` environment variable is checked.
- `user_domain_id` - (Optional) The domain ID where the user is located. If omitted, the `OS_USER_DOMAIN_ID` environment variable is checked.
- `project_domain_name` - (Optional) The domain name where the project is located. If omitted, the `OS_PROJECT_DOMAIN_NAME` environment variable is checked.
- `project_domain_id` - (Optional) The domain ID where the project is located If omitted, the `OS_PROJECT_DOMAIN_ID` environment variable is checked.
- `domain_id` - (Optional) The ID of the Domain to scope to (Identity v3). If omitted, the `OS_DOMAIN_ID` environment variable is checked.
- `domain_name` - (Optional) The Name of the Domain to scope to (Identity v3). If omitted, the following environment variables are checked (in this order): `OS_DOMAIN_NAME` .
- `default_domain` - (Optional) The ID of the Domain to scope to if no other domain is specified (Identity v3). If omitted, the environment variable `OS_DEFAULT_DOMAIN` is checked or a default value of "default" will be used.
- `insecure` - (Optional) Trust self-signed SSL certificates. If omitted, the `OS_INSECURE` environment variable is used.
- `cacert_file` - (Optional) Specify a custom CA certificate when communicating over SSL. You can specify either a path to the file or the contents of the certificate. If omitted, the `OS_CACERT` environment variable is used.
- `cert` - (Optional) Specify client certificate file for SSL client authentication. You can specify either a path to the file or the contents of the certificate. If omitted the `OS_CERT` environment variable is used.
- `key` - (Optional) Specify client private key file for SSL client authentication. You can specify either a path to the file or the contents of the key. If omitted the `OS_KEY` environment variable is used.
- `endpoint_type` - (Optional) Specify which type of endpoint to use from the service catalog. It can be set using the OS_ENDPOINT_TYPE environment variable. If not set, public endpoints is used.
- `endpoint_overrides` - (Optional) A set of key/value pairs that can override an endpoint for a specified OpenStack service. Setting an override requires you to specify the full and complete endpoint URL. This might also invalidate any region you have set, too. Please see below for more details. Please use this at your own risk.

- `swauth` - (Optional) Set to `true` to authenticate against Swauth, a Swift-native authentication system. If omitted, the `OS_SWAUTH` environment variable is used. You must also set `username` to the Swauth/Swift username such as `username:project`. Set the `password` to the Swauth/Swift key. Finally, set `auth_url` as the location of the Swift service. Note that this will only work when used with the OpenStack Object Storage resources.
- `use_octavia` - (Optional) If set to `true`, API requests will go the Load Balancer service (Octavia) instead of the Networking service (Neutron).
- `disable_no_cache_header` - (Optional) If set to `true`, the HTTP `Cache-Control: no-cache` header will not be added by default to all API requests. If omitted this header is added to all API requests to force HTTP caches (if any) to go upstream instead of serving cached responses.
- `delayed_auth` - (Optional) If set to `true`, OpenStack authorization will be perfomed, when the service provider client is called.

Overriding Service API Endpoints

There might be a situation in which you want or need to override an API endpoint rather than use the endpoint which was returned to you in the service catalog. You can do this by configuring the `endpoint_overrides` argument in the provider configuration:

```
provider "openstack" {

  endpoint_overrides = {
    "network"  = "https://example.com:9696/v2.0/"
    "volumev2" = "https://volumes.example.com:8776/v2/3eb25ae78e7b42d68276e9bca66c8e44/"
  }

}
```

Note how each URL ends in a "/" and the `volumev2` service includes the tenant/project UUID. You must make sure you specify the full and complete endpoint URL for this to work.

The service keys are the standard service entries used in the OpenStack Identity/Keystone service catalog. This provider supports:

- `compute` : Compute / Nova v2
- `container-infra` : Container Infra / Magnum v1
- `database` : Database / Trove v1
- `dns` : DNS / Designate v2
- `identity` : Identity / Keystone v3
- `image` : Image / Glance v2
- `network` : Networking / Neutron v2
- `object-store` : Object Storage / Swift v1
- `octavia` : Load Balancing as a Service / Octavia v2

- `sharev2` : Shared Filesystem / Manila v2
- `volume` : Block Storage / Cinder v1
- `volumev2` : Block Storage / Cinder v2
- `volumev3` : Block Storage / Cinder v3

Please use this feature at your own risk. If you are unsure about needing to override an endpoint, you most likely do not need to override one.

Additional Logging

This provider has the ability to log all HTTP requests and responses between Terraform and the OpenStack cloud which is useful for troubleshooting and debugging.

To enable these logs, set the `OS_DEBUG` environment variable to `1` along with the usual `TF_LOG=DEBUG` environment variable:

```
$ OS_DEBUG=1 TF_LOG=DEBUG terraform apply
```

If you submit these logs with a bug report, please ensure any sensitive information has been scrubbed first!

OpenStack Releases and Versions

This provider aims to support "vanilla" OpenStack. This means that we do all testing and development using the official upstream OpenStack code. If your OpenStack environment has patches or modifications, we do our best to accommodate these modifications, but we can't guarantee this.

We try to support *all* releases of OpenStack when we can. If your OpenStack cloud is running an older release, we still should be able to support it.

Rackspace Compatibility

Using this OpenStack provider with Rackspace is not supported and not guaranteed to work; however, users have reported success with the following notes in mind:

- Interacting with instances has been seen to work. Interacting with all other resources is either untested or known to not work.
- Use your *password* instead of your Rackspace API KEY.
- Explicitly define the public and private networks in your instances as shown below:

```
resource "openstack_compute_instance_v2" "my_instance" {
  name      = "my_instance"
  region    = "DFW"
  image_id  = "fabe045f-43f8-4991-9e6c-5cabd617538c"
  flavor_id = "general1-4"
  key_pair  = "provisioning_key"

  network {
    uuid = "00000000-0000-0000-0000-000000000000"
    name = "public"
  }

  network {
    uuid = "11111111-1111-1111-1111-111111111111"
    name = "private"
  }
}
```

If you try using this provider with Rackspace and run into bugs, you are welcomed to open a bug report / issue on Github, but please keep in mind that this is unsupported and the reported bug may not be able to be fixed.

If you have successfully used this provider with Rackspace and can add any additional comments, please let us know.

Testing and Development

Thank you for your interest in further developing the OpenStack provider! Here are a few notes which should help you get started. If you have any questions or feel these notes need further details, please open an Issue and let us know.

Coding and Style

This provider aims to adhere to the coding style and practices used in the other major Terraform Providers. However, this is not a strict rule.

We're very mindful that not everyone is a full-time developer (most of the OpenStack Provider contributors aren't!) and we're happy to provide guidance. Don't be afraid if this is your first contribution to the OpenStack provider or even your first contribution to an open source project!

Testing Environment

In order to start fixing bugs or adding features, you need access to an OpenStack environment. If it is safe to do, you can use a production OpenStack cloud which you have access to. However, it's usually safer to work in a development cloud.

DevStack (https://docs.openstack.org/devstack/latest/) is a quick and easy way to spin up an OpenStack cloud. All OpenStack services have DevStack plugins so you can build a DevStack environment to test everything from Nova/Compute to Designate/DNS.

Fully configuring a DevStack installation is outside the scope of this document; however, we'll try to provide assistance where we can.

Gophercloud

This provider uses Gophercloud (https://github.com/gophercloud/gophercloud) as the Go OpenStack SDK. All API interaction between this provider and an OpenStack cloud is done exclusively with Gophercloud.

Adding a Feature

If you'd like to add a new feature to this provider, it must first be supported in Gophercloud. If Gophercloud is missing the feature, then it'll first have to be added there before you can start working on the feature in Terraform. Fortunately, most of the regular OpenStack Provider contributors also work on Gophercloud, so we can try to get the feature added quickly.

If the feature is already included in Gophercloud, then you can begin work directly in the OpenStack provider.

If you have any questions about whether Gophercloud currently supports a certain feature, please feel free to ask and we can verify for you.

Fixing a Bug

Similarly, if you find a bug in this provider, the bug might actually be a Gophercloud bug. If this is the case, then we'll need to get the bug fixed in Gophercloud first.

However, if the bug is with Terraform itself, then you can begin work directly in the OpenStack provider.

Again, if you have any questions about whether the bug you're trying to fix is a Gophercloud but, please ask.

Vendoring

If you require pulling in changes from an external package, such as Gophercloud, this provider uses Go Modules (https://github.com/golang/go/wiki/Modules).

Acceptance Tests

Acceptance Tests are a crucial part of adding features or fixing a bug. Please make sure to read the core testing (https://www.terraform.io/docs/extend/testing/index.html) documentation for more information about how Acceptance Tests work.

In order to run the Acceptance Tests, you'll need to set the following environment variables:

- `OS_IMAGE_ID` or `OS_IMAGE_NAME` - a UUID or name of an existing image in Glance.
- `OS_FLAVOR_ID` or `OS_FLAVOR_NAME` - an ID or name of an existing flavor.
- `OS_POOL_NAME` - The name of a Floating IP pool. In DevStack, this is called `public` and you should set this value to the word `public`.
- `OS_NETWORK_ID` - The UUID of the private network in your test environment. In DevStack, this network is called `private` and you should set this value to the UUID of the `private` network.
- `OS_EXTGW_ID` - The UUID of the public network in your test environment. In DevStack, this network is called `public` and you should set this value to the UUID of the `public` network.

The following additional environment variables might be required depending on the feature or bug you're testing:

- `OS_DB_ENVIRONMENT` - Required if you're working on the `openstack_db_*` resources. Set this value to "1" to enable testing these resources.
- `OS_DB_DATASTORE_VERSION` - Required if you need to set a Trove/Database datastore version.
- `OS_DB_DATASTORE_TYPE` - Required if you need to set a Trove/Database datastore type.
- `OS_DNS_ENVIRONMENT` - Required if you're working on the `openstack_dns_*` resources. Set this value to "1" to enable testing these resources.
- `OS_SWIFT_ENVIRONMENT` - Required if you're working on an `openstack_objectstorage_*` resource. Set this value to "1" to enable testing these resources.
- `OS_LB_ENVIRONMENT` - Required if you're working on the `openstack_lb_*` resources. Set this value to "1" to enable testing these resources.
- `OS_FW_ENVIRONMENT` - Required if you're working on the `openstack_fw_*` resources. Set this value to "1" to enable testing these resources.
- `OS_VPN_ENVIRONMENT` - Required if your'e working on the `openstack_vpn_*` resources. Set this value to "1" to enable testing these resources.
- `OS_SFS_ENVIRONMENT` - Required if your'e working on the `openstack_openstack_sharedfilesystem_*` resources. Set this value to "1" to enable testing these resources.

We recommend only running the acceptance tests related to the feature or bug you're working on. To do this, run:

```
$ cd $GOPATH/src/github.com/terraform-providers/terraform-provider-openstack
$ make testacc TEST=./openstack TESTARGS="-run=<keyword> -count=1"
```

Where `<keyword>` is the full name or partial name of a test. For example:

```
$ make testacc TEST=./openstack TESTARGS="-run=TestAccComputeV2Keypair_basic -count=1"
```

We recommend running tests with logging set to `DEBUG`:

```
$ TF_LOG=DEBUG make testacc TEST=./openstack TESTARGS="-run=TestAccComputeV2Keypair_basic -count=1"
```

And you can even enable OpenStack debugging to see the actual HTTP API requests:

```
$ TF_LOG=DEBUG OS_DEBUG=1 make testacc TEST=./openstack TESTARGS="-run=TestAccComputeV2Keypair_basic -cou
nt=1"
```

Creating a Pull Request

When you're ready to submit a Pull Request, create a branch, commit your code, and push to your forked version of `terraform-provider-openstack`:

```
$ git remote add my-github-username https://github.com/my-github-username/terraform-provider-openstack
$ git checkout -b my-feature
$ git add .
$ git commit
$ git push -u my-github-username my-feature
```

Then navigate to https://github.com/terraform-providers/terraform-provider-openstack (https://github.com/terraform-providers/terraform-provider-openstack) and create a Pull Request.

OpenLab Testing

Once you have created a Pull Request, it will automatically be tested by OpenLab (https://openlabtesting.org/). OpenLab will run most of the Acceptance Tests in a clean OpenStack cloud (see below for the resources which you must tell OpenLab to run). Testing will take between 90-120 minutes and you will receive a notification with a test report when testing has finished.

If there were any failures, check the provided logs.

There are a few reasons for test failures:

1. Your code changes worked in your environment but are not working in a different OpenStack environment.
2. Your code changes caused another test to fail.
3. OpenLab is having issues.

If you are unable to determine why the failures happened, please ask and we'll look into the cause.

The OpenLab integration has a few keywords that you can use to retest your code. Simply make a comment in your Pull Request with one of the following:

- `recheck` - Run the standard test suite again.
- `recheck designate` - Run the tests for the `openstack_dns_*` resources.
- `recheck trove` - Run the tests for the `openstack_db_*` resources.
- `recheck lbaas` - Run the tests for the `openstack_lb_*` resources.
- `recheck fwaas` - Run the tests for the `openstack_fw_*` resources.
- `recheck stable/mitaka` - Run the standard test suite on OpenStack Mitaka.
- `recheck stable/newton` - Run the standard test suite on OpenStack Newton.
- `recheck stable/ocata` - Run the standard test suite on OpenStack Ocata.
- `recheck stable/pike` - Run the standard test suite on OpenStack Pike.
- `recheck stable/queens` - Run the standard test suite on OpenStack Queens.

openstack_blockstorage_availability_zones_v3

Use this data source to get a list of Block Storage availability zones from OpenStack

Example Usage

```
data "openstack_blockstorage_availability_zones_v3" "zones" {}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the Block Storage client. If omitted, the `region` argument of the provider is used.
- `state` - (Optional) The `state` of the availability zones to match. Can either be `available` or `unavailable`. Default is `available`.

Attributes Reference

`id` is set to hash of the returned zone list. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `state` - See Argument Reference above.
- `names` - The names of the availability zones, ordered alphanumerically, that match the queried `state`.

openstack_blockstorage_snapshot_v2

Use this data source to get information about an existing snapshot.

Example Usage

```
data "openstack_blockstorage_snapshot_v2" "snapshot_1" {
  name        = "snapshot_1"
  most_recent = true
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Block Storage client. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the snapshot.
- `status` - (Optional) The status of the snapshot.
- `volume_id` - (Optional) The ID of the snapshot's volume.
- `most_recent` - (Optional) Pick the most recently created snapshot if there are multiple results.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `status` - See Argument Reference above.
- `volume_id` - See Argument Reference above.
- `description` - The snapshot's description.
- `size` - The size of the snapshot.
- `metadata` - The snapshot's metadata.

openstack_blockstorage_snapshot_v3

Use this data source to get information about an existing snapshot.

Example Usage

```
data "openstack_blockstorage_snapshot_v3" "snapshot_1" {
  name        = "snapshot_1"
  most_recent = true
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V3 Block Storage client. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the snapshot.
- `status` - (Optional) The status of the snapshot.
- `volume_id` - (Optional) The ID of the snapshot's volume.
- `most_recent` - (Optional) Pick the most recently created snapshot if there are multiple results.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `status` - See Argument Reference above.
- `volume_id` - See Argument Reference above.
- `description` - The snapshot's description.
- `size` - The size of the snapshot.
- `metadata` - The snapshot's metadata.

openstack_compute_availability_zones_v2

Use this data source to get a list of availability zones from OpenStack

Example Usage

```
data "openstack_compute_availability_zones_v2" "zones" {}
```

Argument Reference

- `region` - (Optional) The `region` to fetch availability zones from, defaults to the provider's `region`
- `state` - (Optional) The `state` of the availability zones to match, default ("available").

Attributes Reference

`id` is set to hash of the returned zone list. In addition, the following attributes are exported:

- `names` - The names of the availability zones, ordered alphanumerically, that match the queried `state`

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.
- `flavor_id` - (Optional) The ID of the flavor. Conflicts with the `name`, `min_ram` and `min_disk`
- `name` - (Optional) The name of the flavor. Conflicts with the `flavor_id`.
- `min_ram` - (Optional) The minimum amount of RAM (in megabytes). Conflicts with the `flavor_id`.
- `ram` - (Optional) The exact amount of RAM (in megabytes).
- `min_disk` - (Optional) The minimum amount of disk (in gigabytes). Conflicts with the `flavor_id`.
- `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:

- `extra_specs` - Key/Value pairs of metadata for the flavor.
- `is_public` - Whether the flavor is public or private.

openstack_compute_keypair_v2

Use this data source to get the ID and public key of an OpenStack keypair.

Example Usage

```
data "openstack_compute_keypair_v2" "kp" {
  name = "sand"
}
```

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` - (Required) The unique name of the keypair.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `fingerprint` - The fingerprint of the OpenSSH key.
- `public_key` - The OpenSSH-formatted public key of the keypair.

openstack_containerinfra_clustertemplate_v1

Use this data source to get the ID of an available OpenStack Magnum cluster template.

Example Usage

```
data "openstack_containerinfra_clustertemplate_v1" "clustertemplate_1" {
  name = "clustertemplate_1"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 Container Infra client. If omitted, the `region` argument of the provider is used.
- `name` - (Required) The name of the cluster template.

Attributes Reference

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

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `project_id` - The project of the cluster template.
- `user_id` - The user of the cluster template.
- `created_at` - The time at which cluster template was created.
- `updated_at` - The time at which cluster template was updated.
- `apiserver_port` - The API server port for the Container Orchestration Engine for this cluster template.
- `coe` - The Container Orchestration Engine for this cluster template.
- `cluster_distro` - The distro for the cluster (fedora-atomic, coreos, etc.).
- `dns_nameserver` - Address of the DNS nameserver that is used in nodes of the cluster.
- `docker_storage_driver` - Docker storage driver. Changing this updates the Docker storage driver of the existing cluster template.
- `docker_volume_size` - The size (in GB) of the Docker volume.
- `external_network_id` - The ID of the external network that will be used for the cluster.

- `fixed_network` - The fixed network that will be attached to the cluster.
- `fixed_subnet` - =The fixed subnet that will be attached to the cluster.
- `flavor` - The flavor for the nodes of the cluster.
- `master_flavor` - The flavor for the master nodes.
- `floating_ip_enabled` - Indicates whether created cluster should create IP floating IP for every node or not.
- `http_proxy` - The address of a proxy for receiving all HTTP requests and relay them.
- `https_proxy` - The address of a proxy for receiving all HTTPS requests and relay them.
- `image` - The reference to an image that is used for nodes of the cluster.
- `insecure_registry` - The insecure registry URL for the cluster template.
- `keypair_id` - The name of the Compute service SSH keypair.
- `labels` - The list of key value pairs representing additional properties of the cluster template.
- `master_lb_enabled` - Indicates whether created cluster should has a loadbalancer for master nodes or not.
- `network_driver` - The name of the driver for the container network.
- `no_proxy` - A comma-separated list of IP addresses that shouldn't be used in the cluster.
- `public` - Indicates whether cluster template should be public.
- `registry_enabled` - Indicates whether Docker registry is enabled in the cluster.
- `server_type` - The server type for the cluster template.
- `tls_disabled` - Indicates whether the TLS should be disabled in the cluster.
- `volume_driver` - The name of the driver that is used for the volumes of the cluster nodes.

openstack_containerinfra_cluster_v1

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

Example Usage

```
data "openstack_containerinfra_cluster_v1" "cluster_1" {
  name = "cluster_1"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 Container Infra client. If omitted, the `region` argument of the provider is used.
- `name` - (Required) The name of the cluster.

Attributes Reference

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

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `project_id` - The project of the cluster.
- `user_id` - The user of the cluster.
- `created_at` - The time at which cluster was created.
- `updated_at` - The time at which cluster was updated.
- `api_address` - COE API address.
- `coe_version` - COE software version.
- `cluster_template_id` - The UUID of the V1 Container Infra cluster template.
- `create_timeout` - The timeout (in minutes) for creating the cluster.
- `discovery_url` - The URL used for cluster node discovery.
- `docker_volume_size` - The size (in GB) of the Docker volume.
- `flavor` - The flavor for the nodes of the cluster.
- `master_flavor` - The flavor for the master nodes.

- `keypair` - The name of the Compute service SSH keypair.
- `labels` - The list of key value pairs representing additional properties of the cluster.
- `master_count` - The number of master nodes for the cluster.
- `node_count` - The number of nodes for the cluster.
- `master_addresses` - IP addresses of the master node of the cluster.
- `node_addresses` - IP addresses of the node of the cluster.
- `stack_id` - UUID of the Orchestration service stack.

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_fw_policy_v1

Use this data source to get firewall policy information of an available OpenStack firewall policy.

Example Usage

```
data "openstack_fw_policy_v1" "policy" {
  name = "tf_test_policy"
}
```

Argument Reference

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

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `policy_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `description` - The description of the firewall policy.
- `audited` - The audit status of the firewall policy.
- `shared` - The sharing status of the firewall policy.
- `rules` - The array of one or more firewall rules that comprise the policy.

openstack_identity_auth_scope_v3

Use this data source to get authentication information about the current auth scope in use. This can be used as self-discovery or introspection of the username or project name currently in use.

Example Usage

```
data "openstack_identity_auth_scope_v3" "scope" {
  name = "my_scope"
}
```

Argument Reference

- `name` - (Required) The name of the scope. This is an arbitrary name which is only used as a unique identifier so an actual token isn't used as the ID.
- `region` - (Optional) The region in which to obtain the V3 Identity client. A Identity client is needed to retrieve tokens IDs. If omitted, the `region` argument of the provider is used.

Attributes Reference

`id` is set to the name given to the scope. In addition, the following attributes are exported:

- `user_name` - The username of the scope.
- `user_id` - The user ID the of the scope.
- `user_domain_name` - The domain name of the user.
- `user_domain_id` - The domain ID of the user.
- `domain_name` - The domain name of the scope.
- `domain_id` - The domain ID of the scope.
- `project_name` - The project name of the scope.
- `project_id` - The project ID of the scope.
- `project_domain_name` - The domain name of the project.
- `project_domain_id` - The domain ID of the project.
- `roles` - A list of roles in the current scope. See reference below.

The `roles` block contains:

- `role_id` - The ID of the role.

- `role_name` - The name of the role.

openstack_identity_endpoint_v3

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

> **Note:** This usually requires admin privileges.

Example Usage

```
data "openstack_identity_endpoint_v3" "endpoint_1" {
  service_name = "demo"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the endpoint.
- `endpoint_region` - (Optional) The region the endpoint is assigned to. The `region` and `endpoint_region` can be different.
- `service_id` - (Optional) The service id this endpoint belongs to.
- `service_name` - (Optional) The service name of the endpoint.
- `service_type` - (Optional) The service type of the endpoint.
- `interface` - (Optional) The endpoint interface. Valid values are `public`, `internal`, and `admin`. Default value is `public`

Attributes Reference

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

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `endpoint_region` - See Argument Reference above.
- `service_id` - See Argument Reference above.
- `service_name` - See Argument Reference above.
- `service_type` - See Argument Reference above.

- `interface` - See Argument Reference above.
- `url` - The endpoint URL.

openstack_identity_group_v3

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

Note: This usually requires admin privileges.

Example Usage

```
data "openstack_identity_group_v3" "admins" {
  name = "admins"
}
```

Argument Reference

- `name` - The name of the group.
- `domain_id` - (Optional) The domain the group 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 group. In addition, the following attributes are exported:

- `name` - See Argument Reference above.
- `domain_id` - See Argument Reference above.
- `region` - See Argument Reference above.
- `description` - A description of the group.

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_service_v3

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

Note: This usually requires admin privileges.

Example Usage

```
data "openstack_identity_service_v3" "service_1" {
  name = "keystone"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The service name.
- `type` - (Optional) The service type.
- `enabled` - (Optional) The service status.

Attributes Reference

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

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `type` - See Argument Reference above.
- `enabled` - See Argument Reference above.
- `description` - The service description.

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:

- `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 (https://developer.openstack.org/api-ref/identity/v3/#list-users) 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.
- `description` - A description of the user.

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 https://docs.openstack.org/glance/latest/user/metadefs-concepts.html (https://docs.openstack.org/glance/latest/user/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` - The tags list of the image.
- `updated_at` - The date the image was last updated.

openstack_keymanager_container_v1

Use this data source to get the ID of an available Barbican container.

Example Usage

```
data "openstack_keymanager_container_v1" "example" {
  name = "my_container"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 KeyManager client. A KeyManager client is needed to fetch a container. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The Container name.

Attributes Reference

The following attributes are exported:

- `container_ref` - The container reference / where to find the container.
- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `type` - The container type.
- `secret_refs` - A set of dictionaries containing references to secrets. The structure is described below.
- `creator_id` - The creator of the container.
- `status` - The status of the container.
- `created_at` - The date the container was created.
- `updated_at` - The date the container was last updated.
- `consumers` - The list of the container consumers. The structure is described below.

The `secret_refs` block supports:

- `name` - The name of the secret reference. The reference names must correspond the container type, more details are available here (https://docs.openstack.org/barbican/stein/api/reference/containers.html).
- `secret_ref` - The secret reference / where to find the secret, URL.

The `consumers` block supports:

- `name` - The name of the consumer.
- `url` - The consumer URL.

openstack_keymanager_secret_v1

Use this data source to get the ID and the payload of an available Barbican secret

Important Security Notice The payload of this data source will be stored *unencrypted* in your Terraform state file. **Use of this resource for production deployments is *not* recommended**.

Example Usage

```
data "openstack_keymanager_secret_v1" "example" {
  mode        = "cbc"
  secret_type = "passphrase"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 KeyManager client. A KeyManager client is needed to fetch a secret. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The Secret name.
- `bit_length` - (Optional) The Secret bit length.
- `algorithm` - (Optional) The Secret algorithm.
- `mode` - (Optional) The Secret mode.
- `secret_type` - (Optional) The Secret type. For more information see Secret types (https://docs.openstack.org/barbican/latest/api/reference/secret_types.html).
- `acl_only` - (Optional) Select the Secret with an ACL that contains the user. Project scope is ignored. Defaults to `false`.
- `expiration_filter` - (Optional) Date filter to select the Secret with expiration matching the specified criteria. See Date Filters below for more detail.
- `created_at_filter` - (Optional) Date filter to select the Secret with created matching the specified criteria. See Date Filters below for more detail.
- `updated_at_filter` - (Optional) Date filter to select the Secret with updated matching the specified criteria. See Date Filters below for more detail.

Date Filters

The values for the `expiration_filter`, `created_at_filter`, and `updated_at_filter` parameters are comma-separated lists of time stamps in RFC3339 format. The time stamps can be prefixed with any of these comparison operators: *gt:* (greater-than), *gte:* (greater-than-or-equal), *lt:* (less-than), *lte:* (less-than-or-equal).

For example, to get a passphrase a Secret with CBC moda, that will expire in January of 2020:

```
data "openstack_keymanager_secret_v1" "date_filter_example" {
  mode              = "cbc"
  secret_type       = "passphrase"
  expiration_filter = "gt:2020-01-01T00:00:00Z"
}
```

Attributes Reference

The following attributes are exported:

- `secret_ref` - The secret reference / where to find the secret.
- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `bit_length` - See Argument Reference above.
- `algorithm` - See Argument Reference above.
- `mode` - See Argument Reference above.
- `secret_type` - See Argument Reference above.
- `acl_only` - See Argument Reference above.
- `expiration_filter` - See Argument Reference above.
- `created_at_filter` - See Argument Reference above.
- `updated_at_filter` - See Argument Reference above.
- `payload` - The secret payload.
- `payload_content_type` - The Secret content type.
- `payload_content_encoding` - The Secret encoding.
- `content_types` - The map of the content types, assigned on the secret.
- `creator_id` - The creator of the secret.
- `status` - The status of the secret.
- `expiration` - The date the secret will expire.
- `created_at` - The date the secret was created.
- `updated_at` - The date the secret was last updated.

- `metadata` - The map of metadata, assigned on the secret, which has been explicitly and implicitly added.

openstack_networking_addressscope_v2

Use this data source to get the ID of an available OpenStack address-scope.

Example Usage

```
data "openstack_networking_addressscope_v2" "public_addressscope" {
  name       = "public_addressscope"
  shared     = true
  ip_version = 4
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve address-scopes. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) Name of the address-scope.
- `ip_version` - (Optional) IP version.
- `shared` - (Optional) Indicates whether this address-scope is shared across all projects.
- `project_id` - (Optional) The owner of the address-scope.

Attributes Reference

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

- `name` - See Argument Reference above.
- `ip_version` - See Argument Reference above.
- `shared` - See Argument Reference above.
- `project_id` - See Argument Reference above.

openstack_networking_floatingip_v2

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

Example Usage

```
data "openstack_networking_floatingip_v2" "floatingip_1" {
  address = "192.168.0.4"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve floating IP ids. If omitted, the `region` argument of the provider is used.
- `description` - (Optional) Human-readable description of the floating IP.
- `address` - (Optional) The IP address of the floating IP.
- `pool` - (Optional) The name of the pool from which the floating IP belongs to.
- `port_id` - (Optional) The ID of the port the floating IP is attached.
- `status` - status of the floating IP (ACTIVE/DOWN).
- `fixed_ip` - (Optional) The specific IP address of the internal port which should be associated with the floating IP.
- `tags` - (Optional) The list of floating IP tags to filter.
- `tenant_id` - (Optional) The owner of the floating IP.

Attributes Reference

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

- `all_tags` - A set of string tags applied on the floating IP.
- `dns_name` - The floating IP DNS name. Available, when Neutron DNS extension is enabled.
- `dns_domain` - The floating IP DNS domain. Available, when Neutron DNS extension is enabled.

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.
- `description` - (Optional) Human-readable description of the network.
- `status` - (Optional) The status of the network.
- `external` - (Optional) The external routing facility 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.
- `transparent_vlan` - (Optional) The VLAN transparent attribute for the network.
- `tags` - (Optional) The list of network tags to filter.
- `mtu` - (Optional) The network MTU to filter. Available, when Neutron `net-mtu` extension is enabled.

Attributes Reference

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

- `admin_state_up` - The administrative state of the network.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `region` - See Argument Reference above.
- `external` - See Argument Reference above.

- `shared` - Specifies whether the network resource can be accessed by any tenant or not.
- `availability_zone_hints` - The availability zone candidates for the network.
- `transparent_vlan` - See Argument Reference above.
- `all_tags` - The set of string tags applied on the network.
- `mtu` - See Argument Reference above.
- `dns_domain` - The network DNS domain. Available, when Neutron DNS extension is enabled

openstack_networking_port_ids_v2

Use this data source to get a list of Openstack Port IDs matching the specified criteria.

Example Usage

```
data "openstack_networking_port_ids_v2" "ports" {
  name = "port"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve port ids. If omitted, the `region` argument of the provider is used.
- `project_id` - (Optional) The owner of the port.
- `name` - (Optional) The name of the port.
- `description` - (Optional) Human-readable description of the port.
- `admin_state_up` - (Optional) The administrative state of the port.
- `network_id` - (Optional) The ID of the network the port belongs to.
- `device_owner` - (Optional) The device owner of the port.
- `mac_address` - (Optional) The MAC address of the port.
- `device_id` - (Optional) The ID of the device the port belongs to.
- `fixed_ip` - (Optional) The port IP address filter.
- `status` - (Optional) The status of the port.
- `security_group_ids` - (Optional) The list of port security group IDs to filter.
- `tags` - (Optional) The list of port tags to filter.
- `sort_key` - (Optional) Sort ports based on a certain key. Defaults to none.
- `sort_direction` - (Optional) Order the results in either `asc` or `desc`. Defaults to none.

Attributes Reference

`ids` is set to the list of Openstack Port IDs.

openstack_networking_port_v2

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

Example Usage

```
data "openstack_networking_port_v2" "port_1" {
  name = "port_1"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve port ids. If omitted, the `region` argument of the provider is used.
- `project_id` - (Optional) The owner of the port.
- `port_id` - (Optional) The ID of the port.
- `name` - (Optional) The name of the port.
- `description` - (Optional) Human-readable description of the port.
- `admin_state_up` - (Optional) The administrative state of the port.
- `network_id` - (Optional) The ID of the network the port belongs to.
- `device_owner` - (Optional) The device owner of the port.
- `mac_address` - (Optional) The MAC address of the port.
- `device_id` - (Optional) The ID of the device the port belongs to.
- `fixed_ip` - (Optional) The port IP address filter.
- `status` - (Optional) The status of the port.
- `security_group_ids` - (Optional) The list of port security group IDs to filter.
- `tags` - (Optional) The list of port tags to filter.
- `dns_name` - (Optional) The port DNS name to filter. Available, when Neutron DNS extension is enabled.

Attributes Reference

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

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

- `port_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `network_id` - See Argument Reference above.
- `device_owner` - See Argument Reference above.
- `mac_address` - See Argument Reference above.
- `device_id` - See Argument Reference above.
- `allowed_address_pairs` - An IP/MAC Address pair of additional IP addresses that can be active on this port. The structure is described below.
- `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 set of security group IDs applied on the port.
- `all_tags` - The set of string tags applied on the port.
- `extra_dhcp_option` - An extra DHCP option configured on the port. The structure is described below.
- `binding` - The port binding information. The structure is described below.
- `dns_name` - See Argument Reference above.
- `dns_assignment` - The list of maps representing port DNS assignments.

The `allowed_address_pairs` attribute has fields below:

- `ip_address` - The additional IP address.
- `mac_address` - The additional MAC address.

The `extra_dhcp_option` attribute has fields below:

- `name` - Name of the DHCP option.
- `value` - Value of the DHCP option.
- `ip_version` - IP protocol version

The `binding` attribute has fields below:

- `host_id` - The ID of the host, which has the allocatee port.
- `profile` - A JSON string containing the binding profile information.
- `vnic_type` - VNIC type for the port.
- `vif_details` - A map of JSON strings containing additional details for this specific binding.
- `vif_type` - The VNIC type of the port binding.

openstack_networking_qos_bandwidth_limit_rule_v2

Use this data source to get the ID of an available OpenStack QoS bandwidth limit rule.

Example Usage

```
data "openstack_networking_qos_bandwidth_limit_rule_v2" "qos_bandwidth_limit_rule_1" {
  max_kbps = 300
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a Neutron QoS bandwidth limit rule. If omitted, the `region` argument of the provider is used.
- `qos_policy_id` - (Required) The QoS policy reference.
- `max_kbps` - (Optional) The maximum kilobits per second of a QoS bandwidth limit rule.
- `max_burst_kbps` - (Optional) The maximum burst size in kilobits of a QoS bandwidth limit rule.
- `direction` - (Optional) The direction of traffic.

Attributes Reference

`id` is set to the `qos_policy_id/bandwidth_limit_rule_id` format of the found QoS bandwidth limit rule. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `qos_policy_id` - See Argument Reference above.
- `max_kbps` - See Argument Reference above.
- `max_burst_kbps` - See Argument Reference above.
- `direction` - See Argument Reference above.

openstack_networking_qos_dscp_marking_rule_v2

Use this data source to get the ID of an available OpenStack QoS DSCP marking rule.

Example Usage

```
data "openstack_networking_qos_dscp_marking_rule_v2" "qos_dscp_marking_rule_1" {
  dscp_mark = 26
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a Neutron QoS DSCP marking rule. If omitted, the `region` argument of the provider is used.
- `qos_policy_id` - (Required) The QoS policy reference.
- `dscp_mark` - (Optional) The value of a DSCP mark.

Attributes Reference

`id` is set to the `qos_policy_id/dscp_marking_rule_id` format of the found QoS DSCP marking rule. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `qos_policy_id` - See Argument Reference above.
- `dscp_mark` - See Argument Reference above.

openstack_networking_qos_minimum_bandwidth_rule_v2

Use this data source to get the ID of an available OpenStack QoS minimum bandwidth rule.

Example Usage

```
data "openstack_networking_qos_minimum_bandwidth_rule_v2" "qos_min_bw_rule_1" {
  min_kbps = 2000
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to create a Neutron QoS minimum bandwidth rule. If omitted, the `region` argument of the provider is used.
- `qos_policy_id` - (Required) The QoS policy reference.
- `min_kbps` - (Optional) The value of a minimum kbps bandwidth.

Attributes Reference

`id` is set to the `qos_policy_id/minimum_bandwidth_rule_id` format of the found QoS minimum bandwidth rule. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `qos_policy_id` - See Argument Reference above.
- `min_kbps` - See Argument Reference above.

openstack_networking_qos_policy_v2

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

Example Usage

```
data "openstack_networking_qos_policy_v2" "qos_policy_1" {
  name = "qos_policy_1"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Networking client. A Networking client is needed to retrieve a QoS policy ID. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the QoS policy.
- `project_id` - (Optional) The owner of the QoS policy.
- `shared` - (Optional) Whether this QoS policy is shared across all projects.
- `description` - (Optional) The human-readable description for the QoS policy.
- `is_default` - (Optional) Whether the QoS policy is default policy or not.
- `tags` - (Optional) The list of QoS policy tags to filter.

Attributes Reference

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

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `created_at` - The time at which QoS policy was created.
- `updated_at` - The time at which QoS policy was created.
- `shared` - See Argument Reference above.
- `description` - See Argument Reference above.
- `is_default` - See Argument Reference above.
- `revision_number` - The revision number of the QoS policy.
- `all_tags` - The set of string tags applied on the QoS policy.

openstack_networking_router_v2

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

Example Usage

```
data "openstack_networking_router_v2" "router" {
  name = "router_1"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve router ids. If omitted, the `region` argument of the provider is used.
- `router_id` - (Optional) The UUID of the router resource.
- `name` - (Optional) The name of the router.
- `description` - (Optional) Human-readable description of the router.
- `admin_state_up` - (Optional) Administrative up/down status for the router (must be "true" or "false" if provided).
- `distributed` - (Optional) Indicates whether or not to get a distributed router.
- `status` - (Optional) The status of the router (ACTIVE/DOWN).
- `tags` - (Optional) The list of router tags to filter.
- `tenant_id` - (Optional) The owner of the router.

Attributes Reference

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

- `enable_snat` - The value that points out if the Source NAT is enabled on the router.
- `external_network_id` - The network UUID of an external gateway for the router.
- `availability_zone_hints` - The availability zone that is used to make router resources highly available.
- `external_fixed_ip` - The external fixed IPs of the router.

The `external_fixed_ip` block supports:

- `subnet_id` - Subnet in which the fixed IP belongs to.
- `ip_address` - The IP address to set on the router.
- `all_tags` - The set of string tags applied on the router.

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.
- `description` - (Optional) Human-readable description the the subnet.
- `tags` - (Optional) The list of security group tags to filter.
- `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` - See Argument Reference above.
- `all_tags` - The set of string tags applied on the security group.
- `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.
- `tags` - (Optional) The list of subnetpool tags to filter.

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.
- `all_tags` - The set of string tags applied on the subnetpool.

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.
- `description` - (Optional) Human-readable description for 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.
- `tags` - (Optional) The list of subnet tags to filter.

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.
- `all_tags` - A set of string tags applied on the subnet.

openstack_networking_trunk_v2

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

Example Usage

```
data "openstack_networking_trunk_v2" "trunk_1" {
  name = "trunk_1"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Neutron client. A Neutron client is needed to retrieve trunk ids. If omitted, the `region` argument of the provider is used.
- `project_id` - (Optional) The owner of the trunk.
- `trunk_id` - (Optional) The ID of the trunk.
- `name` - (Optional) The name of the trunk.
- `description` - (Optional) Human-readable description of the trunk.
- `port_id` - (Optional) The ID of the trunk parent port.
- `admin_state_up` - (Optional) The administrative state of the trunk.
- `status` - (Optional) The status of the trunk.
- `tags` - (Optional) The list of trunk tags to filter.

Attributes Reference

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

- `all_tags` - The set of string tags applied on the trunk.
- `sub_port` - The set of the trunk subports. The structure of each subport is described below.

The `sub_port` attribute has fields below:

- `port_id` - The ID of the trunk subport.
- `segmentation_type` - The segmenation tecnology used, e.g., "vlan".
- `segmentation_id` - The numeric id of the subport segment.

openstack_sharedfilesystem_availability_zones_v2

Use this data source to get a list of Shared File System availability zones from OpenStack

Example Usage

```
data "openstack_sharedfilesystem_availability_zones_v2" "zones" {}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Shared File System client. If omitted, the `region` argument of the provider is used.

Attributes Reference

`id` is set to hash of the returned zone list. In addition, the following attributes are exported:

- `region` - See Argument Reference above.
- `names` - The names of the availability zones, ordered alphanumerically.

openstack_sharedfilesystem_sharenetwork_v2

Use this data source to get the ID of an available Shared File System share network.

Example Usage

```
data "openstack_sharedfilesystem_sharenetwork_v2" "sharenetwork_1" {
  name = "sharenetwork_1"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Shared File System client. A Shared File System client is needed to read a share network. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The name of the share network.
- `description` - (Optional) The human-readable description of the share network.
- `project_id` - (Optional) The owner of the share network.
- `neutron_net_id` - (Optional) The neutron network UUID of the share network.
- `neutron_subnet_id` - (Optional) The neutron subnet UUID of the share network.
- `security_service_id` - (Optional) The security service IDs associated with the share network.
- `network_type` - (Optional) The share network type. Can either be VLAN, VXLAN, GRE, or flat.
- `segmentation_id` - (Optional) The share network segmentation ID.
- `cidr` - (Optional) The share network CIDR.
- `ip_version` - (Optional) The IP version of the share network. Can either be 4 or 6.

Attributes Reference

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

- `region` - See Argument Reference above.
- `project_id` - The owner of the Share Network.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `neutron_net_id` - See Argument Reference above.
- `neutron_subnet_id` - See Argument Reference above.

- `security_service_id` - See Argument Reference above.
- `network_type` - See Argument Reference above.
- `segmentation_id` - See Argument Reference above.
- `cidr` - See Argument Reference above.
- `ip_version` - See Argument Reference above.
- `security_service_ids` - The list of security service IDs associated with the share network.

openstack_sharedfilesystem_share_v2

Use this data source to get the ID of an available Shared File System share.

Example Usage

```
data "openstack_sharedfilesystem_share_v2" "share_1" {
  name = "share_1"
}
```

Argument Reference

- `name` - (Optional) The name of the share.
- `description` - (Optional) The human-readable description for the share.
- `project_id` - (Optional) The owner of the share.
- `snapshot_id` - (Optional) The UUID of the share's base snapshot.
- `share_network_id` - (Optional) The UUID of the share's share network.
- `export_location_path` - (Optional) The export location path of the share. Available since Manila API version 2.35.
- `metadata` - (Optional) One or more metadata key and value pairs as a dictionary of strings.
- `status` - (Optional) A share status filter. A valid value is `creating`, `error`, `available`, `deleting`, `error_deleting`, `manage_starting`, `manage_error`, `unmanage_starting`, `unmanage_error`, `unmanaged`, `extending`, `extending_error`, `shrinking`, `shrinking_error`, or `shrinking_possible_data_loss_error`.
- `is_public` - (Optional) The level of visibility for the share. length.

Attributes Reference

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

- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `snapshot_id` - See Argument Reference above.
- `share_network_id` - See Argument Reference above.
- `export_location_path` - See Argument Reference above.
- `metadata` - See Argument Reference above.

- `status` - See Argument Reference above.
- `is_public` - See Argument Reference above.
- `region` - The region in which to obtain the V2 Shared File System client.
- `availability_zone` - The share availability zone.
- `share_proto` - The share protocol.
- `size` - The share size, in GBs.
- `export_locations` - A list of export locations. For example, when a share server has more than one network interface, it can have multiple export locations.

openstack_sharedfilesystem_snapshot_v2

Use this data source to get the ID of an available Shared File System snapshot.

Example Usage

```
data "openstack_sharedfilesystem_snapshot_v2" "snapshot_1" {
  name = "snapshot_1"
}
```

Argument Reference

- `region` - (Optional) The region in which to obtain the V2 Shared File System client.
- `name` - (Optional) The name of the snapshot.
- `description` - (Optional) The human-readable description of the snapshot.
- `project_id` - (Optional) The owner of the snapshot.
- `status` - (Optional) A snapshot status filter. A valid value is `available`, `error`, `creating`, `deleting`, `manage_starting`, `manage_error`, `unmanage_starting`, `unmanage_error` or `error_deleting`.

Attributes Reference

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

- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `status` - See Argument Reference above.
- `size` - The snapshot size, in GBs.
- `share_id` - The UUID of the source share that was used to create the snapshot.
- `share_proto` - The file system protocol of a share snapshot.
- `share_size` - The share snapshot size, in GBs.

openstack_blockstorage_quotaset_v2

Manages a V2 block storage quotaset resource within OpenStack.

Note: This usually requires admin privileges.

Note: This resource has a no-op deletion so no actual actions will be done against the OpenStack API in case of delete call.

Example Usage

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

resource "openstack_blockstorage_quotaset_v2" "quotaset_1" {
  project_id = "${openstack_identity_project_v3.project_1.id}"
  volumes   = 10
  snapshots = 4
  gigabytes = 100
  per_volume_gigabytes = 10
  backups = 4
  backup_gigabytes = 10
  groups = 100
}
```

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 quotaset.
- `project_id` - (Required) ID of the project to manage quotas. Changing this creates a new quotaset.
- `volumes` - (Optional) Quota value for volumes. Changing this updates the existing quotaset.
- `snapshots` - (Optional) Quota value for snapshots. Changing this updates the existing quotaset.
- `gigabytes` - (Optional) Quota value for gigabytes. Changing this updates the existing quotaset.
- `per_volume_gigabytes` - (Optional) Quota value for gigabytes per volume . Changing this updates the existing quotaset.
- `backups` - (Optional) Quota value for backups. Changing this updates the existing quotaset.
- `backup_gigabytes` - (Optional) Quota value for backup gigabytes. Changing this updates the existing quotaset.
- `groups` - (Optional) Quota value for groups. Changing this updates the existing quotaset.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `volumes` - See Argument Reference above.
- `snapshots` - See Argument Reference above.
- `gigabytes` - See Argument Reference above.
- `per_volume_gigabytes` - See Argument Reference above.
- `backups` - See Argument Reference above.
- `backup_gigabytes` - See Argument Reference above.
- `groups` - See Argument Reference above.

Import

Quotasets can be imported using the `project_id`, e.g.

```
$ terraform import openstack_blockstorage_quotaset_v2.quotaset_1 2a0f2240-c5e6-41de-896d-e80d97428d6b
```

openstack_blockstorage_quotaset_v3

Manages a V3 block storage quotaset resource within OpenStack.

Note: This usually requires admin privileges.

Note: This resource has a no-op deletion so no actual actions will be done against the OpenStack API in case of delete call.

Example Usage

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

resource "openstack_blockstorage_quotaset_v3" "quotaset_1" {
  project_id = "${openstack_identity_project_v3.project_1.id}"
  volumes   = 10
  snapshots = 4
  gigabytes = 100
  per_volume_gigabytes = 10
  backups = 4
  backup_gigabytes = 10
  groups = 100
}
```

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 quotaset.
- `project_id` - (Required) ID of the project to manage quotas. Changing this creates a new quotaset.
- `volumes` - (Optional) Quota value for volumes. Changing this updates the existing quotaset.
- `snapshots` - (Optional) Quota value for snapshots. Changing this updates the existing quotaset.
- `gigabytes` - (Optional) Quota value for gigabytes. Changing this updates the existing quotaset.
- `per_volume_gigabytes` - (Optional) Quota value for gigabytes per volume . Changing this updates the existing quotaset.
- `backups` - (Optional) Quota value for backups. Changing this updates the existing quotaset.
- `backup_gigabytes` - (Optional) Quota value for backup gigabytes. Changing this updates the existing quotaset.
- `groups` - (Optional) Quota value for groups. Changing this updates the existing quotaset.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `volumes` - See Argument Reference above.
- `snapshots` - See Argument Reference above.
- `gigabytes` - See Argument Reference above.
- `per_volume_gigabytes` - See Argument Reference above.
- `backups` - See Argument Reference above.
- `backup_gigabytes` - See Argument Reference above.
- `groups` - See Argument Reference above.

Import

Quotasets can be imported using the `project_id`, e.g.

```
$ terraform import openstack_blockstorage_quotaset_v3.quotaset_1 2a0f2240-c5e6-41de-896d-e80d97428d6b
```

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 default --op new
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n node.session.aut
h.authmethod -v ${self.data.auth_method}
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n node.session.aut
h.username -v ${self.data.auth_username}
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n node.session.aut
h.password -v ${self.data.auth_password}
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 node.startup -v
automatic
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_blockstorage_volume_attach_v3

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) v3 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_v3" "volume_1" {
  name = "volume_1"
  size = 1
}

resource "openstack_blockstorage_volume_attach_v3" "va_1" {
  volume_id  = "${openstack_blockstorage_volume_v3.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 V3 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 default --op new
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n node.session.aut
h.authmethod -v ${self.data.auth_method}
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n node.session.aut
h.username -v ${self.data.auth_username}
iscsiadm -m node -T ${self.data.target_iqn} -p ${self.data.target_portal} --op update -n node.session.aut
h.password -v ${self.data.auth_password}
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 node.startup -v
automatic
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_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-eeb1-4c10-8d33-26f0409a755d
```

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-eeb1-4c10-8d33-26f0409a755d
```

openstack_blockstorage_volume_v3

Manages a V3 volume resource within OpenStack.

Example Usage

```
resource "openstack_blockstorage_volume_v3" "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).
- `enable_online_resize` - (Optional) When this option is set it allows extending attached volumes. Note: updating size of an attached volume requires Cinder support for version 3.42 and a compatible storage driver.
- `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.
- `multiattach` - (Optional) Allow the volume to be attached to more than one Compute instance.

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.
- `multiattach` - See Argument Reference above.

Import

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

```
$ terraform import openstack_blockstorage_volume_v3.volume_1 ea257959-eeb1-4c10-8d33-26f0409a755d
```

openstack_compute_flavor_access_v2

Manages a project access for flavor V2 resource within OpenStack.

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

Example Usage

```
resource "openstack_identity_project_v3" "project_1" {
  name = "my-project"
}

resource "openstack_compute_flavor_v2" "flavor_1" {
  name      = "my-flavor"
  ram       = "8096"
  vcpus     = "2"
  disk      = "20"
  is_public = false
}

resource "openstack_compute_flavor_access_v2" "access_1" {
  tenant_id = "${openstack_identity_project_v3.project_1.id}"
  flavor_id = "${openstack_compute_flavor_v2.flavor_1.id}"
}
```

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 flavor access.
- `flavor_id` - (Required) The UUID of flavor to use. Changing this creates a new flavor access.
- `tenant_id` - (Required) The UUID of tenant which is allowed to use the flavor. Changing this creates a new flavor access.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `flavor_id` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.

Import

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

```
$ terraform import openstack_compute_flavor_access_v2.access_1 <flavor_id>/<tenant_id>
```

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-5affc7af1106
```

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.
- `wait_until_associated` - (Optional) In cases where the OpenStack environment does not automatically wait until the association has finished, set this option to have Terraform poll the instance until the floating IP has been associated. Defaults to false.

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_id>/<fixed_ip>
```

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` (/docs/providers/openstack/r/networking_floatingip_v2.html) 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-e2b959ede9d2
```

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" {
  instance_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.example.com"

  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 or ids 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 (https://docs.openstack.org/nova/latest/user/block-device-mapping.html) 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.
- `power_state` - (Optional) Provide the VM state. Only 'active' and 'shutoff' are supported values. *Note*: If the initial power_state is the shutoff the VM will be stopped immediately after build and the provisioners like remote-exec or files are not supported.
- `tags` - (Optional) A set of string tags for the instance. Changing this updates the existing instance tags.
- `vendor_options` - (Optional) Map of additional vendor-specific options. Supported options are described below.

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.
- `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.
- `device_type` - (Optional) The low-level device type that will be used. Most common thing is to leave this empty. Changing this creates a new server.
- `disk_bus` - (Optional) The low-level disk bus that will be used. Most common thing is to leave this empty. 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. The query must use the `JsonFilter` syntax which is described here (https://docs.openstack.org/nova/latest/admin/configuration/schedulers.html#jsonfilter). At this time, only simple queries are supported. Compound queries using `and`, `or`, or `not` are not supported. An example of a simple query is:

  ```
  [">=", "$free_ram_mb", "1024"]
  ```

- `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.
- `content` - (Required) The contents of the file. Limited to 255 bytes.

The `vendor_options` block supports:

- `ignore_resize_confirmation` - (Optional) Boolean to control whether to ignore manual confirmation of the instance resizing. This can be helpful to work with some OpenStack clouds which automatically confirm resizing of instances after some timeout.

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.
- `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.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the instance, which have been explicitly and implicitly added.

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 (https://review.openstack.org/#/c/126309/) 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",
    ]
  }
}
```

Instances and Networks

Instances almost always require a network. Here are some notes to be aware of with how Instances and Networks relate:

- In scenarios where you only have one network available, you can create an instance without specifying a `network` block. OpenStack will automatically launch the instance on this network.
- If you have access to more than one network, you will need to specify a network with a `network` block. Not specifying a network will result in the following error:

```
* openstack_compute_instance_v2.instance: Error creating OpenStack server:
Expected HTTP response code [201 202] when accessing [POST https://example.com:8774/v2.1/servers], but go
t 409 instead
{"conflictingRequest": {"message": "Multiple possible networks found, use a Network ID to be more specifi
c.", "code": 409}}
```

- If you intend to use the `openstack_compute_interface_attach_v2` resource, you still need to make sure one of the above points is satisfied. An instance cannot be created without a valid network configuration even if you intend to use `openstack_compute_interface_attach_v2` after the instance has been created.

Importing instances

Importing instances can be tricky, since the nova api does not offer all information provided at creation time for later retrieval. Network interface attachment order, and number and sizes of ephemeral disks are examples of this.

Importing basic instance

Assume you want to import an instance with one ephemeral root disk, and one network interface.

Your configuration would look like the following:

```
resource "openstack_compute_instance_v2" "basic_instance" {
  name            = "basic"
  flavor_id       = "<flavor_id>"
  key_pair        = "<keyname>"
  security_groups = ["default"]
  image_id =  "<image_id>"

  network {
    name = "<network_name>"
  }
}
```

Then you execute `terraform import openstack_compute_instance_v2.basic_instance <instance_id>`

Importing an instance with multiple emphemeral disks

The importer cannot read the emphemeral disk configuration of an instance, so just specify image_id as in the configuration of the basic instance example.

Importing instance with multiple network interfaces.

Nova returns the network interfaces grouped by network, thus not in creation order. That means that if you have multiple network interfaces you must take care of the order of networks in your configuration.

As example we want to import an instance with one ephemeral root disk, and 3 network interfaces.

Examples

```
resource "openstack_compute_instance_v2" "boot-from-volume" {
  name            = "boot-from-volume"
  flavor_id       = "<flavor_id"
  key_pair        = "<keyname>"
  image_id        = <image_id>
  security_groups = ["default"]

  network {
    name = "<network1>"
  }
  network {
    name = "<netowork2>"
  }
  network {
    name = "<network1>"
    fixed_ip_v4 = "<fixed_ip_v4>"
  }

}
```

In the above configuration the networks are out of order compared to what nova and thus the import code returns, which means the plan will not be empty after import.

So either with care check the plan and modify configuration, or read the network order in the state file after import and modify your configuration accordingly.

- A note on ports. If you have created a neutron port independent of an instance, then the import code has no way to detect that the port is created idenpendently, and therefore on deletion of imported instances you might have port resources in your project, which you expected to be created by the instance and thus to also be deleted with the instance.

Importing instances with multiple block storage volumes.

We have an instance with two block storage volumes, one bootable and one non-bootable. Note that we only configure the bootable device as block_device. The other volumes can be specified as `openstack_blockstorage_volume_v2`

```
resource "openstack_compute_instance_v2" "instance_2" {
  name            = "instance_2"
  image_id        = "<image_id>"
  flavor_id       = "<flavor_id>"
  key_pair        = "<keyname>"
  security_groups = ["default"]

  block_device {
    uuid = <image_id>"
    source_type = "image"
    destination_type = "volume"
    boot_index = 0
    delete_on_termination = true
  }

   network {
    name = "<network_name>"
  }
}
resource "openstack_blockstorage_volume_v2" "volume_1" {
  size        = 1
  name =     "<vol_name>"
}
resource "openstack_compute_volume_attach_v2" "va_1" {
  volume_id = "${openstack_blockstorage_volume_v2.volume_1.id}"
  instance_id = "${openstack_compute_instance_v2.instance_2.id}"
}
```

To import the instance outlined in the above configuration do the following:

```
terraform import openstack_compute_instance_v2.instance_2 <instance_id>
import openstack_blockstorage_volume_v2.volume_1 <volume_id>
terraform import openstack_compute_volume_attach_v2.va_1
<instance_id>/<volume_id>
```

- A note on block storage volumes, the importer does not read delete_on_termination flag, and always assumes true. If you import an instance created with delete_on_termination false, you end up with "orphaned" volumes after destruction of instances.

openstack_compute_interface_attach_v2

Attaches a Network Interface (a Port) to an Instance using the OpenStack Compute (Nova) v2 API.

Example Usage

Basic Attachment

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

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

resource "openstack_compute_interface_attach_v2" "ai_1" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  network_id  = "${openstack_networking_port_v2.network_1.id}"
}
```

Attachment Specifying a Fixed IP

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

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

resource "openstack_compute_interface_attach_v2" "ai_1" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  network_id  = "${openstack_networking_port_v2.network_1.id}"
  fixed_ip    = "10.0.10.10"
}
```

Attachment Using an Existing Port

```
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"
}


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

resource "openstack_compute_interface_attach_v2" "ai_1" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  port_id     = "${openstack_networking_port_v2.port_1.id}"
}
```

Attaching Multiple Interfaces

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

resource "openstack_networking_port_v2" "ports" {
  count          = 2
  name           = "${format("port-%02d", count.index + 1)}"
  network_id     = "${openstack_networking_network_v2.network_1.id}"
  admin_state_up = "true"
}

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

resource "openstack_compute_interface_attach_v2" "attachments" {
  count       = 2
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  port_id     = "${openstack_networking_port_v2.ports.*.id[count.index]}"
}
```

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

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

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

resource "openstack_networking_port_v2" "ports" {
  count          = 2
  name           = "${format("port-%02d", count.index + 1)}"
  network_id     = "${openstack_networking_network_v2.network_1.id}"
  admin_state_up = "true"
}

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

resource "openstack_compute_interface_attach_v2" "ai_1" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  port_id     = "${openstack_networking_port_v2.ports.*.id[0]}"
}

resource "openstack_compute_interface_attach_v2" "ai_2" {
  instance_id = "${openstack_compute_instance_v2.instance_1.id}"
  port_id     = "${openstack_networking_port_v2.ports.*.id[1]}"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to create the interface attachment. If omitted, the `region` argument of the provider is used. Changing this creates a new attachment.
- `instance_id` - (Required) The ID of the Instance to attach the Port or Network to.
- `port_id` - (Optional) The ID of the Port to attach to an Instance. *NOTE*: This option and `network_id` are mutually exclusive.
- `network_id` - (Optional) The ID of the Network to attach to an Instance. A port will be created automatically. *NOTE*: This option and `port_id` are mutually exclusive.
- `fixed_ip` - (Optional) An IP address to assosciate with the port. *NOTE*: This option cannot be used with port_id. You must specifiy a network_id. The IP address must lie in a range on the supplied network.

Attributes Reference

The following attributes are exported:

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

- `port_id` - See Argument Reference above.
- `network_id` - See Argument Reference above.
- `fixed_ip` - See Argument Reference above.

Import

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

```
$ terraform import openstack_compute_interface_attach_v2.ai_1 89c60255-9bd6-460c-822a-e2b959ede9d2/45670584-225f-46c3-b33e-6707b589b666
```

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 AAAAB3NzaC1yc2EAAAADAQABAAABAQDAjpC1hwiOCCmKEWxJ4qzTTsJbKzndLotBCz5PcwtUnflmU+gHJ
tWMZKpuEGVi29h0A/+ydKek1O18k10Ff+4tyFjiHDQAnOfgWf7+b1yK+qDip3X1C0UPMbwHlTfSGWLGZqd9LvEFx9k3h/M+VtMvwR1lJ9
LUyTAImnNjWG7TaIPmui30HvM2UiFEmqkr4ijq45MyX2+fLIePLRIF61p4whjHAQYufqyno3BS48icQb4p6iVEZPo4AE2o9oIyQvj2mx4
dk5Y8CgSETOZTYDOR3rU2fZTRDRgPJDH9FWvQjF5tA0p3d9CoWWd2s6GKKbfoUIi8R/Db1BSPJwkqB"
}
```

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_quotaset_v2

Manages a V2 compute quotaset resource within OpenStack.

Note: This usually requires admin privileges.

Note: This resource has a no-op deletion so no actual actions will be done against the OpenStack API in case of delete call.

Example Usage

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

resource "openstack_compute_quotaset_v2" "quotaset_1" {
  project_id           = "${openstack_identity_project_v3.project_1.id}"
  key_pairs            = 10
  ram                  = 40960
  cores                = 32
  instances            = 20
  server_groups        = 4
  server_group_members = 8
}
```

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 quotaset.
- `project_id` - (Required) ID of the project to manage quotas. Changing this creates a new quotaset.
- `fixed_ips` - (Optional) Quota value for fixed IPs. Changing this updates the existing quotaset.
- `floating_ips` - (Optional) Quota value for floating IPs. Changing this updates the existing quotaset.
- `injected_file_content_bytes` - (Optional) Quota value for content bytes of injected files. Changing this updates the existing quotaset.
- `injected_file_path_bytes` - (Optional) Quota value for path bytes of injected files. Changing this updates the existing quotaset.
- `injected_files` - (Optional) Quota value for injected files. Changing this updates the existing quotaset.
- `key_pairs` - (Optional) Quota value for key pairs. Changing this updates the existing quotaset.

- `metadata_items` - (Optional) Quota value for metadata items. Changing this updates the existing quotaset.
- `ram` - (Optional) Quota value for RAM. Changing this updates the existing quotaset.
- `security_group_rules` - (Optional) Quota value for security group rules. Changing this updates the existing quotaset.
- `security_groups` - (Optional) Quota value for security groups. Changing this updates the existing quotaset.
- `cores` - (Optional) Quota value for cores. Changing this updates the existing quotaset.
- `instances` - (Optional) Quota value for instances. Changing this updates the existing quotaset.
- `server_groups` - (Optional) Quota value for server groups. Changing this updates the existing quotaset.
- `server_group_members` - (Optional) Quota value for server groups members. Changing this updates the existing quotaset.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `fixed_ips` - See Argument Reference above.
- `floating_ips` - See Argument Reference above.
- `injected_file_content_bytes` - See Argument Reference above.
- `injected_file_path_bytes` - See Argument Reference above.
- `injected_files` - See Argument Reference above.
- `key_pairs` - See Argument Reference above.
- `metadata_items` - See Argument Reference above.
- `ram` - See Argument Reference above.
- `security_group_rules` - See Argument Reference above.
- `security_groups` - See Argument Reference above.
- `cores` - See Argument Reference above.
- `instances` - See Argument Reference above.
- `server_groups` - See Argument Reference above.
- `server_group_members` - See Argument Reference above.

Import

Quotasets can be imported using the `project_id`, e.g.

```
$ terraform import openstack_compute_quotaset_v2.quotaset_1 2a0f2240-c5e6-41de-896d-e80d97428d6b
```

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` (/docs/providers/openstack/r/networking_secgroup_v2.html) and `openstack_networking_secgroup_rule_v2` (/docs/providers/openstack/r/networking_secgroup_rule_v2.html) 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 (https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol#Control_messages).

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-bdd5-7f1e663f5edf
```

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. All policies 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.
- `soft-affinity` - All instances/servers launched in this group will be hosted on the same compute node if possible, but if not possible they still will be scheduled instead of failure. To use this policy your OpenStack environment should support Compute service API 2.15 or above.
- `soft-anti-affinity` - All instances/servers launched in this group will be hosted on different compute nodes if possible, but if not possible they still will be scheduled instead of failure. To use this policy your OpenStack environment should support Compute service API 2.15 or above.

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-bdd5-7f1e663f5edf
```

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 = "${openstack_compute_instance_v2.instance_1.id}"
  volume_id   = "${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}"
}
```

Using Multiattach-enabled volumes

Multiattach Volumes are dependent upon your OpenStack cloud and not all clouds support multiattach.

```
resource "openstack_blockstorage_volume_v3" "volume_1" {
  name        = "volume_1"
  size        = 1
  multiattach = true
}

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

resource "openstack_compute_instance_v2" "instance_2" {
  name            = "instance_2"
  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}"
  multiattach = true
}

resource "openstack_compute_volume_attach_v2" "va_2" {
  instance_id = "${openstack_compute_instance_v2.instance_2.id}"
  volume_id   = "${openstack_blockstorage_volume_v2.volume_1.id}"
  multiattach = true

  depends_on = ["openstack_compute_volume_attach_v2.va_1"]
}
```

It is recommended to use `depends_on` for the attach resources to enforce the volume attachments to happen one at a time.

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.
- `multiattach` - (Optional) Enable attachment of multiattach-capable volumes.

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.
- `multiattach` - See Argument Reference above.

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-e2b959ede9d2/45670584-225f-46c3-b33e-6707b589b666
```

openstack_containerinfra_clustertemplate_v1

Manages a V1 Magnum cluster template resource within OpenStack.

Example Usage

Create a Cluster template

```
resource "openstack_containerinfra_clustertemplate_v1" "clustertemplate_1" {
  name                  = "clustertemplate_1"
  image                 = "Fedora-Atomic-27"
  coe                   = "kubernetes"
  flavor                = "m1.small"
  master_flavor         = "m1.medium"
  dns_nameserver        = "1.1.1.1"
  docker_storage_driver = "devicemapper"
  docker_volume_size    = 10
  volume_driver         = "cinder"
  network_driver        = "flannel"
  server_type           = "vm"
  master_lb_enabled     = true
  floating_ip_enabled   = false

  labels = {
    kube_tag                        = "1.11.1"
    kube_dashboard_enabled          = "true"
    prometheus_monitoring           = "true"
    influx_grafana_dashboard_enabled = "true"
  }
}
```

Argument reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 Container Infra client. A Container Infra client is needed to create a cluster template. If omitted,the `region` argument of the provider is used. Changing this creates a new cluster template.
- `name` - (Required) The name of the cluster template. Changing this updates the name of the existing cluster template.
- `project_id` - (Optional) The project of the cluster template. Required if admin wants to create a cluster template in another project. Changing this creates a new cluster template.
- `user_id` - (Optional) The user of the cluster template. Required if admin wants to create a cluster template for another user. Changing this creates a new cluster template.
- `apiserver_port` - (Optional) The API server port for the Container Orchestration Engine for this cluster template. Changing this updates the API server port of the existing cluster template.

- `coe` - (Required) The Container Orchestration Engine for this cluster template. Changing this updates the engine of the existing cluster template.
- `cluster_distro` - (Optional) The distro for the cluster (fedora-atomic, coreos, etc.). Changing this updates the cluster distro of the existing cluster template.
- `dns_nameserver` - (Optional) Address of the DNS nameserver that is used in nodes of the cluster. Changing this updates the DNS nameserver of the existing cluster template.
- `docker_storage_driver` - (Optional) Docker storage driver. Changing this updates the Docker storage driver of the existing cluster template.
- `docker_volume_size` - (Optional) The size (in GB) of the Docker volume. Changing this updates the Docker volume size of the existing cluster template.
- `external_network_id` - (Optional) The ID of the external network that will be used for the cluster. Changing this updates the external network ID of the existing cluster template.
- `fixed_network` - (Optional) The fixed network that will be attached to the cluster. Changing this updates the fixed network of the existing cluster template.
- `fixed_subnet` - (Optional) The fixed subnet that will be attached to the cluster. Changing this updates the fixed subnet of the existing cluster template.
- `flavor` - (Optional) The flavor for the nodes of the cluster. Can be set via the `OS_MAGNUM_FLAVOR` environment variable. Changing this updates the flavor of the existing cluster template.
- `master_flavor` - (Optional) The flavor for the master nodes. Can be set via the `OS_MAGNUM_MASTER_FLAVOR` environment variable. Changing this updates the master flavor of the existing cluster template.
- `floating_ip_enabled` - (Optional) Indicates whether created cluster should create floating IP for every node or not. Changing this updates the floating IP enabled attribute of the existing cluster template.
- `http_proxy` - (Optional) The address of a proxy for receiving all HTTP requests and relay them. Changing this updates the HTTP proxy address of the existing cluster template.
- `https_proxy` - (Optional) The address of a proxy for receiving all HTTPS requests and relay them. Changing this updates the HTTPS proxy address of the existing cluster template.
- `image` - (Required) The reference to an image that is used for nodes of the cluster. Can be set via the `OS_MAGNUM_IMAGE` environment variable. Changing this updates the image attribute of the existing cluster template.
- `insecure_registry` - (Optional) The insecure registry URL for the cluster template. Changing this updates the insecure registry attribute of the existing cluster template.
- `keypair_id` - (Optional) The name of the Compute service SSH keypair. Changing this updates the keypair of the existing cluster template.
- `labels` - (Optional) The list of key value pairs representing additional properties of the cluster template. Changing this updates the labels of the existing cluster template.
- `master_lb_enabled` - (Optional) Indicates whether created cluster should has a loadbalancer for master nodes or not. Changing this updates the attribute of the existing cluster template.

- `network_driver` - (Optional) The name of the driver for the container network. Changing this updates the network driver of the existing cluster template.
- `no_proxy` - (Optional) A comma-separated list of IP addresses that shouldn't be used in the cluster. Changing this updates the no proxy list of the existing cluster template.
- `public` - (Optional) Indicates whether cluster template should be public. Changing this updates the public attribute of the existing cluster template.
- `registry_enabled` - (Optional) Indicates whether Docker registry is enabled in the cluster. Changing this updates the registry enabled attribute of the existing cluster template.
- `server_type` - (Optional) The server type for the cluster template. Changing this updates the server type of the existing cluster template.
- `tls_disabled` - (Optional) Indicates whether the TLS should be disabled in the cluster. Changing this updates the attribute of the existing cluster.
- `volume_driver` - (Optional) The name of the driver that is used for the volumes of the cluster nodes. Changing this updates the volume driver of the existing cluster template.

Attributes reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `created_at` - The time at which cluster template was created.
- `updated_at` - The time at which cluster template was created.
- `apiserver_port` - See Argument Reference above.
- `coe` - See Argument Reference above.
- `cluster_distro` - See Argument Reference above.
- `dns_nameserver` - See Argument Reference above.
- `docker_storage_driver` - See Argument Reference above.
- `docker_volume_size` - See Argument Reference above.
- `external_network_id` - See Argument Reference above.
- `fixed_network` - See Argument Reference above.
- `fixed_subnet` - See Argument Reference above.
- `flavor` - See Argument Reference above.
- `master_flavor` - See Argument Reference above.

- `floating_ip_enabled` - See Argument Reference above.
- `http_proxy` - See Argument Reference above.
- `https_proxy` - See Argument Reference above.
- `image` - See Argument Reference above.
- `insecure_registry` - See Argument Reference above.
- `keypair_id` - See Argument Reference above.
- `labels` - See Argument Reference above.
- `links` - A list containing associated cluster template links.
- `master_lb_enabled` - See Argument Reference above.
- `network_driver` - See Argument Reference above.
- `no_proxy` - See Argument Reference above.
- `public` - See Argument Reference above.
- `registry_enabled` - See Argument Reference above.
- `server_type` - See Argument Reference above.
- `tls_disabled` - See Argument Reference above.
- `volume_driver` - See Argument Reference above.

Import

Cluster templates can be imported using the `id`, e.g.

```
$ terraform import openstack_containerinfra_clustertemplate_v1.clustertemplate_1 b9a45c5c-cd03-4958-82aa-b80bf93cb922
```

openstack_containerinfra_cluster_v1

Manages a V1 Magnum cluster resource within OpenStack.

Example Usage

Create a Cluster

```
resource "openstack_containerinfra_cluster_v1" "cluster_1" {
  name                = "cluster_1"
  cluster_template_id = "b9a45c5c-cd03-4958-82aa-b80bf93cb922"
  master_count        = 3
  node_count          = 5
  keypair             = "ssh_keypair"
}
```

Argument reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 Container Infra client. A Container Infra client is needed to create a cluster. If omitted, the `region` argument of the provider is used. Changing this creates a new cluster.
- `name` - (Required) The name of the cluster. Changing this updates the name of the existing cluster template.
- `project_id` - (Optional) The project of the cluster. Required if admin wants to create a cluster in another project. Changing this creates a new cluster.
- `user_id` - (Optional) The user of the cluster. Required if admin wants to create a cluster template for another user. Changing this creates a new cluster.
- `cluster_template_id` - (Required) The UUID of the V1 Container Infra cluster template. Changing this creates a new cluster.
- `create_timeout` - (Optional) The timeout (in minutes) for creating the cluster. Changing this creates a new cluster.
- `discovery_url` - (Optional) The URL used for cluster node discovery. Changing this creates a new cluster.
- `docker_volume_size` - (Optional) The size (in GB) of the Docker volume. Changing this creates a new cluster.
- `flavor` - (Optional) The flavor for the nodes of the cluster. Can be set via the `OS_MAGNUM_FLAVOR` environment variable. Changing this creates a new cluster.
- `master_flavor` - (Optional) The flavor for the master nodes. Can be set via the `OS_MAGNUM_MASTER_FLAVOR` environment variable. Changing this creates a new cluster.
- `keypair` - (Optional) The name of the Compute service SSH keypair. Changing this creates a new cluster.

- `labels` - (Optional) The list of key value pairs representing additional properties of the cluster. Changing this creates a new cluster.
- `master_count` - (Optional) The number of master nodes for the cluster. Changing this creates a new cluster.
- `node_count` - (Optional) The number of nodes for the cluster. Changing this creates a new cluster.

Attributes reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `created_at` - The time at which cluster was created.
- `updated_at` - The time at which cluster was created.
- `api_address` - COE API address.
- `coe_version` - COE software version.
- `cluster_template_id` - See Argument Reference above.
- `container_version` - Container software version.
- `create_timeout` - See Argument Reference above.
- `discovery_url` - See Argument Reference above.
- `docker_volume_size` - See Argument Reference above.
- `flavor` - See Argument Reference above.
- `master_flavor` - See Argument Reference above.
- `keypair` - See Argument Reference above.
- `labels` - See Argument Reference above.
- `master_count` - See Argument Reference above.
- `node_count` - See Argument Reference above.
- `master_addresses` - IP addresses of the master node of the cluster.
- `node_addresses` - IP addresses of the node of the cluster.
- `stack_id` - UUID of the Orchestration service stack.

Import

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

```
$ terraform import openstack_containerinfra_cluster_v1.cluster_1 ce0f9463-dd25-474b-9fe8-94de63e5e42b
```

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_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_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_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_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. *Note:* if an IPv6 address contains brackets (`[ ]`), the brackets will be stripped and the modified address will be recorded in the state.
- `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_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_identity_application_credential_v3

Manages a V3 Application Credential resource within OpenStack Keystone.

Note: All arguments including the application credential name and secret will be stored in the raw state as plain-text. Read more about sensitive data in state (/docs/state/sensitive-data.html).

Note: An Application Credential is created within the authenticated user project scope and is not visible by an admin or other accounts. The Application Credential visibility is similar to `openstack_compute_keypair_v2` (/docs/providers/openstack/r/compute_keypair_v2.html).

Example Usage

Predefined secret

Application credential below will have only one `swiftoperator` role.

```
resource "openstack_identity_application_credential_v3" "swift" {
  name        = "swift"
  description = "Swift technical application credential"
  secret      = "supersecret"
  roles       = ["swiftoperator"]
  expires_at  = "2019-02-13T12:12:12Z"
}
```

Unrestricted with autogenerated secret and unlimited TTL

Application credential below will inherit all the current user's roles.

WARNING: Restrictions on these Identity operations are deliberately imposed as a safeguard to prevent a compromised application credential from regenerating itself. Disabling this restriction poses an inherent added risk.

```
resource "openstack_identity_application_credential_v3" "unrestricted" {
  name         = "unrestricted"
  description  = "Unrestricted application credential"
  unrestricted = true
}

output "application_credential_secret" {
  value = "${openstack_identity_application_credential_v3.unrestricted.secret}"
}
```

Argument Reference

The following arguments are supported:

- `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 application credential.
- `name` - (Required) A name of the application credential. Changing this creates a new application credential.
- `description` - (Optional) A description of the application credential. Changing this creates a new application credential.
- `unrestricted` - (Optional) A flag indicating whether the application credential may be used for creation or destruction of other application credentials or trusts. Changing this creates a new application credential.
- `secret` - (Optional) The secret for the application credential. If omitted, it will be generated by the server. Changing this creates a new application credential.
- `roles` - (Optional) A collection of one or more role names, which this application credential has to be associated with its project. If omitted, all the current user's roles within the scoped project will be inherited by a new application credential. Changing this creates a new application credential.
- `expires_at` - (Optional) The expiration time of the application credential in the RFC3339 timestamp format (e.g. `2019-03-09T12:58:49Z`). If omitted, an application credential will never expire. Changing this creates a new application credential.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `unrestricted` - See Argument Reference above.
- `secret` - See Argument Reference above.
- `roles` - See Argument Reference above.
- `expires_at` - See Argument Reference above.
- `project_id` - The ID of the project the application credential was created for and that authentication requests using this application credential will be scoped to.

Import

Application Credentials can be imported using the `id` , e.g.

```
$ terraform import openstack_identity_application_credential_v3.application_credential_1 c17304b7-0953-47
38-abb0-67005882b0a0
```

openstack_identity_endpoint_v3

Manages a V3 Endpoint resource within OpenStack Keystone.

Note: This usually requires admin privileges.

Example Usage

```
resource "openstack_identity_service_v3" "service_1" {
  name = "my-service"
  type = "my-service-type"
}

resource "openstack_identity_endpoint_v3" "endpoint_1" {
  name = "my-endpoint"
  service_id = "${openstack_identity_service_v3.service_1.id}"
  endpoint_region = "${openstack_identity_service_v3.service_1.region}"
  url = "http://my-endpoint"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used.
- `name` - (Optional) The endpoint name.
- `endpoint_region` - (Required) The endpoint region. The `region` and `endpoint_region` can be different.
- `url` - (Required) The endpoint url.
- `interface` - (Optional) The endpoint interface. Valid values are `public`, `internal` and `admin`. Default value is `public`
- `service_id` - (Required) The endpoint service ID.

Attributes Reference

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

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

- `url` - See Argument Reference above.
- `interface` - See Argument Reference above.
- `service_id` - See Argument Reference above.
- `service_name` - The service name of the endpoint.
- `service_type` - The service type of the endpoint.

Import

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

```
$ terraform import openstack_identity_endpoint_v3.endpoint_1 5392472b-106a-4845-90c6-7c8445f18770
```

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-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_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_service_v3

Manages a V3 Service resource within OpenStack Keystone.

Note: This usually requires admin privileges.

Example Usage

```
resource "openstack_identity_service_v3" "service_1" {
  name = "custom"
  type = "custom"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V3 Keystone client. If omitted, the `region` argument of the provider is used.
- `name` - (Required) The service name.
- `description` - (Optional) The service description.
- `type` - (Required) The service type.
- `enabled` - (Optional) The service status. Defaults to `true`.

Attributes Reference

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

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `type` - See Argument Reference above.
- `enabled` - See Argument Reference above.
- `description` - See Argument Reference above.

Import

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

```
$ terraform import openstack_identity_service_v3.service_1 6688e967-158a-496f-a224-cae3414e6b61
```

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 (https://docs.openstack.org/releasenotes/keystone/ocata.html) 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_access_accept_v2

Manages memberships status for the shared OpenStack Glance V2 Image within the destination project, which has a member proposal.

Example Usage

Accept a shared image membershipship proposal within the current project.

```
data "openstack_images_image_v2" "rancheros" {
  name          = "RancherOS"
  visibility    = "shared"
  member_status = "all"
}

resource "openstack_images_image_access_accept_v2" "rancheros_member" {
  image_id = "${data.openstack_images_image_v2.rancheros.id}"
  status   = "accepted"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Glance client. A Glance client is needed to manage Image memberships. If omitted, the `region` argument of the provider is used. Changing this creates a new membership.
- `image_id` - (Required) The proposed image ID.
- `member_id` - (Optional) The member ID, e.g. the target project ID. Optional for admin accounts. Defaults to the current scope project ID.
- `status` - (Required) The membership proposal status. Can either be `accepted`, `rejected` or `pending`.

Attributes Reference

The following attributes are exported:

- `created_at` - The date the image membership was created.
- `updated_at` - The date the image membership was last updated.
- `schema` - The membership schema.

Import

Image access acceptance status can be imported using the `image_id`, e.g.

```
$ terraform import openstack_images_image_access_accept_v2 89c60255-9bd6-460c-822a-e2b959ede9d2
```

openstack_images_image_access_v2

Manages members for the shared OpenStack Glance V2 Image within the source project, which owns the Image.

Example Usage

Unprivileged user

Create a shared image and propose a membership to the `bed6b6cbb86a4e2d8dc2735c2f1000e4` project ID.

```
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"
  visibility       = "shared"

  properties = {
    key = "value"
  }
}

resource "openstack_images_image_access_v2" "rancheros_member" {
  image_id  = "${openstack_images_image_v2.rancheros.id}"
  member_id = "bed6b6cbb86a4e2d8dc2735c2f1000e4"
}
```

Privileged user

Create a shared image and set a membership to the `bed6b6cbb86a4e2d8dc2735c2f1000e4` project ID.

```
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"
  visibility       = "shared"

  properties = {
    key = "value"
  }
}

resource "openstack_images_image_access_v2" "rancheros_member" {
  image_id  = "${openstack_images_image_v2.rancheros.id}"
  member_id = "bed6b6cbb86a4e2d8dc2735c2f1000e4"
  status    = "accepted"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Glance client. A Glance client is needed to manage Image members. If omitted, the `region` argument of the provider is used. Changing this creates a new resource.
- `image_id` - (Required) The image ID.
- `member_id` - (Required) The member ID, e.g. the target project ID.
- `status` - (Optional) The member proposal status. Optional if admin wants to force the member proposal acceptance. Can either be `accepted`, `rejected` or `pending`. Defaults to `pending`. Foridden for non-admin users.

Attributes Reference

The following attributes are exported:

- `created_at` - The date the image access was created.
- `updated_at` - The date the image access was last updated.
- `schema` - The member schema.

Import

Image access can be imported using the `image_id` and the `member_id`, separated by a slash, e.g.

```
$ terraform import openstack_images_image_access_v2 89c60255-9bd6-460c-822a-e2b959ede9d2/bed6b6cbb86a4e2d8dc2735c2f1000e4
```

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. See the "Notes" section for further information about properties.
- `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.
- `verify_checksum` - (Optional) If false, the checksum will not be verified once the image is finished uploading. Defaults to true.
- `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 https://docs.openstack.org/glance/latest/user/metadefs-concepts.html (https://docs.openstack.org/glance/latest/user/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.
- `updated_at` - The date the image was last updated.
- `update_at` - (**Deprecated** - use `updated_at` instead)
- `visibility` - See Argument Reference above.

Notes

Properties

This resource supports the ability to add properties to a resource during creation as well as add, update, and delete properties during an update of this resource.

Newer versions of OpenStack are adding some read-only properties to each image. These properties start with the prefix `os_` . If these properties are detected, this resource will automatically reconcile these with the user-provided properties.

In addition, the `direct_url` property is also automatically reconciled if the Image Service set it.

Import

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

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

openstack_keymanager_container_v1

Manages a V1 Barbican container resource within OpenStack.

Example Usage

The container with the TLS certificates, which can be used by the loadbalancer HTTPS listener.

```
resource "openstack_keymanager_secret_v1" "certificate_1" {
  name                 = "certificate"
  payload              = "${file("cert.pem")}"
  secret_type          = "certificate"
  payload_content_type = "text/plain"
}

resource "openstack_keymanager_secret_v1" "private_key_1" {
  name                 = "private_key"
  payload              = "${file("cert-key.pem")}"
  secret_type          = "private"
  payload_content_type = "text/plain"
}

resource "openstack_keymanager_secret_v1" "intermediate_1" {
  name                 = "intermediate"
  payload              = "${file("intermediate-ca.pem")}"
  secret_type          = "certificate"
  payload_content_type = "text/plain"
}

resource "openstack_keymanager_container_v1" "tls_1" {
  name = "tls"
  type = "certificate"

  secret_refs {
    name       = "certificate"
    secret_ref = "${openstack_keymanager_secret_v1.certificate_1.secret_ref}"
  }

  secret_refs {
    name       = "private_key"
    secret_ref = "${openstack_keymanager_secret_v1.private_key_1.secret_ref}"
  }

  secret_refs {
    name       = "intermediates"
    secret_ref = "${openstack_keymanager_secret_v1.intermediate_1.secret_ref}"
  }
}

data "openstack_networking_subnet_v2" "subnet_1" {
  name = "my-subnet"
}

resource "openstack_lb_loadbalancer_v2" "lb_1" {
  name          = "loadbalancer"
  vip_subnet_id = "${data.openstack_networking_subnet_v2.subnet_1.id}"
}

resource "openstack_lb_listener_v2" "listener_1" {
  name                      = "https"
  protocol                  = "TERMINATED_HTTPS"
  protocol_port             = 443
  loadbalancer_id           = "${openstack_lb_loadbalancer_v2.lb_1.id}"
  default_tls_container_ref = "${openstack_keymanager_container_v1.tls_1.container_ref}"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 KeyManager client. A KeyManager client is needed to create a container. If omitted, the `region` argument of the provider is used. Changing this creates a new V1 container.
- `name` - (Optional) Human-readable name for the Container. Does not have to be unique.
- `type` - (Required) Used to indicate the type of container. Must be one of `generic`, `rsa` or `certificate`.
- `secret_refs` - (Optional) A set of dictionaries containing references to secrets. The structure is described below.

The `secret_refs` block supports:

- `name` - (Optional) The name of the secret reference. The reference names must correspond the container type, more details are available here (https://docs.openstack.org/barbican/stein/api/reference/containers.html).
- `secret_ref` - (Required) The secret reference / where to find the secret, URL.

Attributes Reference

The following attributes are exported:

- `container_ref` - The container reference / where to find the container.
- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `type` - See Argument Reference above.
- `secret_refs` - See Argument Reference above.
- `creator_id` - The creator of the container.
- `status` - The status of the container.
- `created_at` - The date the container was created.
- `updated_at` - The date the container was last updated.
- `consumers` - The list of the container consumers. The structure is described below.

The `consumers` block supports:

- `name` - The name of the consumer.
- `url` - The consumer URL.

Import

Containers can be imported using the container id (the last part of the container reference), e.g.:

```
$ terraform import openstack_keymanager_container_v1.container_1 0c6cd26a-c012-4d7b-8034-057c0f1c2953
```

openstack_keymanager_secret_v1

Manages a V1 Barbican secret resource within OpenStack.

Important Security Notice The payload of this resource will be stored *unencrypted* in your Terraform state file. **Use of this resource for production deployments is *not* recommended**.

Example Usage

Simple secret

```
resource "openstack_keymanager_secret_v1" "secret_1" {
  algorithm            = "aes"
  bit_length           = 256
  mode                 = "cbc"
  name                 = "mysecret"
  payload              = "foobar"
  payload_content_type = "text/plain"
  secret_type          = "passphrase"

  metadata = {
    key = "foo"
  }
}
```

Secret with whitespaces

```
resource "openstack_keymanager_secret_v1" "secret_1" {
  name                     = "password"
  payload                  = "${base64encode("password with the whitespace at the end ")}"
  secret_type              = "passphrase"
  payload_content_type     = "application/octet-stream"
  payload_content_encoding = "base64"
}
```

Secret with the expiration date

```
resource "openstack_keymanager_secret_v1" "secret_1" {
  name                 = "certificate"
  payload              = "${file("certificate.pem")}"
  secret_type          = "certificate"
  payload_content_type = "text/plain"
  expiration           = "${timeadd(timestamp(), format("%dh", 8760))}" # one year in hours

  lifecycle {
    ignore_changes = [
      expiration
    ]
  }
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V1 KeyManager client. A KeyManager client is needed to create a secret. If omitted, the `region` argument of the provider is used. Changing this creates a new V1 secret.
- `name` - (Optional) Human-readable name for the Secret. Does not have to be unique.
- `bit_length` - (Optional) Metadata provided by a user or system for informational purposes.
- `algorithm` - (Optional) Metadata provided by a user or system for informational purposes.
- `mode` - (Optional) Metadata provided by a user or system for informational purposes.
- `secret_type` - (Optional) Used to indicate the type of secret being stored. For more information see Secret types (https://docs.openstack.org/barbican/latest/api/reference/secret_types.html).
- `payload` - (Optional) The secret's data to be stored. **payload_content_type** must also be supplied if **payload** is included.
- `payload_content_type` - (Optional) (required if **payload** is included) The media type for the content of the payload. Must be one of `text/plain`, `text/plain;charset=utf-8`, `text/plain; charset=utf-8`, `application/octet-stream`, `application/pkcs8`.
- `payload_content_encoding` - (Optional) (required if **payload** is encoded) The encoding used for the payload to be able to include it in the JSON request. Must be either `base64` or `binary`.
- `expiration` - (Optional) The expiration time of the secret in the RFC3339 timestamp format (e.g. `2019-03-09T12:58:49Z`). If omitted, a secret will never expire. Changing this creates a new secret.
- `metadata` - (Optional) Additional Metadata for the secret.

Attributes Reference

The following attributes are exported:

- `secret_ref` - The secret reference / where to find the secret.

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `bit_length` - See Argument Reference above.
- `algorithm` - See Argument Reference above.
- `mode` - See Argument Reference above.
- `secret_type` - See Argument Reference above.
- `payload` - See Argument Reference above.
- `payload_content_type` - See Argument Reference above.
- `payload_content_encoding` - See Argument Reference above.
- `expiration` - See Argument Reference above.
- `content_types` - The map of the content types, assigned on the secret.
- `creator_id` - The creator of the secret.
- `status` - The status of the secret.
- `created_at` - The date the secret was created.
- `updated_at` - The date the secret was last updated.
- `all_metadata` - The map of metadata, assigned on the secret, which has been explicitly and implicitly added.

Import

Secrets can be imported using the secret id (the last part of the secret reference), e.g.:

```
$ terraform import openstack_keymanager_secret_v1.secret_1 8a7a79c2-cf17-4e65-b2ae-ddc8bfcf6c74
```

openstack_lb_l7policy_v2

Manages a Load Balancer L7 Policy 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"
  cidr       = "192.168.199.0/24"
  ip_version = 4
  network_id = "${openstack_networking_network_v2.network_1.id}"
}

resource "openstack_lb_loadbalancer_v2" "loadbalancer_1" {
  name          = "loadbalancer_1"
  vip_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}

resource "openstack_lb_listener_v2" "listener_1" {
  name            = "listener_1"
  protocol        = "HTTP"
  protocol_port   = 8080
  loadbalancer_id = "${openstack_lb_loadbalancer_v2.loadbalancer_1.id}"
}

resource "openstack_lb_pool_v2" "pool_1" {
  name            = "pool_1"
  protocol        = "HTTP"
  lb_method       = "ROUND_ROBIN"
  loadbalancer_id = "${openstack_lb_loadbalancer_v2.loadbalancer_1.id}"
}

resource "openstack_lb_l7policy_v2" "l7policy_1" {
  name             = "test"
  action           = "REDIRECT_TO_POOL"
  description      = "test l7 policy"
  position         = 1
  listener_id      = "${openstack_lb_listener_v2.listener_1.id}"
  redirect_pool_id = "${openstack_lb_pool_v2.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 . If omitted, the `region` argument of the provider is used. Changing this creates a new L7 Policy.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the L7 Policy. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new L7 Policy.
- `name` - (Optional) Human-readable name for the L7 Policy. Does not have to be unique.
- `description` - (Optional) Human-readable description for the L7 Policy.
- `action` - (Required) The L7 Policy action - can either be REDIRECT_TO_POOL, REDIRECT_TO_URL or REJECT.
- `listener_id` - (Required) The Listener on which the L7 Policy will be associated with. Changing this creates a new L7 Policy.
- `position` - (Optional) The position of this policy on the listener. Positions start at 1.
- `redirect_pool_id` - (Optional) Requests matching this policy will be redirected to the pool with this ID. Only valid if action is REDIRECT_TO_POOL.
- `redirect_url` - (Optional) Requests matching this policy will be redirected to this URL. Only valid if action is REDIRECT_TO_URL.
- `admin_state_up` - (Optional) The administrative state of the L7 Policy. A valid value is true (UP) or false (DOWN).

Attributes Reference

The following attributes are exported:

- `id` - The unique ID for the L7 {olicy.
- `region` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `action` - See Argument Reference above.
- `listener_id` - See Argument Reference above.
- `position` - See Argument Reference above.
- `redirect_pool_id` - See Argument Reference above.
- `redirect_url` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.

Import

Load Balancer L7 Policy can be imported using the L7 Policy ID, e.g.:

```
$ terraform import openstack_lb_l7policy_v2.l7policy_1 8a7a79c2-cf17-4e65-b2ae-ddc8bfcf6c74
```

openstack_lb_l7rule_v2

Manages a V2 L7 Rule 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"
  cidr       = "192.168.199.0/24"
  ip_version = 4
  network_id = "${openstack_networking_network_v2.network_1.id}"
}

resource "openstack_lb_loadbalancer_v2" "loadbalancer_1" {
  name          = "loadbalancer_1"
  vip_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}

resource "openstack_lb_listener_v2" "listener_1" {
  name            = "listener_1"
  protocol        = "HTTP"
  protocol_port   = 8080
  loadbalancer_id = "${openstack_lb_loadbalancer_v2.loadbalancer_1.id}"
}

resource "openstack_lb_pool_v2" "pool_1" {
  name            = "pool_1"
  protocol        = "HTTP"
  lb_method       = "ROUND_ROBIN"
  loadbalancer_id = "${openstack_lb_loadbalancer_v2.loadbalancer_1.id}"
}

resource "openstack_lb_l7policy_v2" "l7policy_1" {
  name         = "test"
  action       = "REDIRECT_TO_URL"
  description  = "test description"
  position     = 1
  listener_id  = "${openstack_lb_listener_v2.listener_1.id}"
  redirect_url = "http://www.example.com"
}

resource "openstack_lb_l7rule_v2" "l7rule_1" {
  l7policy_id  = "${openstack_lb_l7policy_v2.l7policy_1.id}"
  type         = "PATH"
  compare_type = "EQUAL_TO"
  value        = "/api"
}
```

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 L7 Rule.
- `tenant_id` - (Optional) Required for admins. The UUID of the tenant who owns the L7 Rule. Only administrative users can specify a tenant UUID other than their own. Changing this creates a new L7 Rule.
- `description` - (Optional) Human-readable description for the L7 Rule.
- `type` - (Required) The L7 Rule type - can either be COOKIE, FILE_TYPE, HEADER, HOST_NAME or PATH.
- `compare_type` - (Required) The comparison type for the L7 rule - can either be CONTAINS, STARTS_WITH, ENDS_WITH, EQUAL_TO or REGEX
- `l7policy_id` - (Required) The ID of the L7 Policy to query. Changing this creates a new L7 Rule.
- `value` - (Required) The value to use for the comparison. For example, the file type to compare.
- `key` - (Optional) The key to use for the comparison. For example, the name of the cookie to evaluate. Valid when `type` is set to COOKIE or HEADER.
- `invert` - (Optional) When true the logic of the rule is inverted. For example, with invert true, equal to would become not equal to. Default is false.
- `admin_state_up` - (Optional) The administrative state of the L7 Rule. A valid value is true (UP) or false (DOWN).

Attributes Reference

The following attributes are exported:

- `id` - The unique ID for the L7 Rule.
- `region` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `type` - See Argument Reference above.
- `compare_type` - See Argument Reference above.
- `l7policy_id` - See Argument Reference above.
- `value` - See Argument Reference above.
- `key` - See Argument Reference above.
- `invert` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `listener_id` - The ID of the Listener owning this resource.

Import

Load Balancer L7 Rule can be imported using the L7 Policy ID and L7 Rule ID separated by a slash, e.g.:

```
$ terraform import openstack_lb_l7rule_v2.l7rule_1 e0bd694a-abbe-450e-b329-0931fd1cc5eb/4086b0c9-b18c-4d1c-b6b8-4c56c3ad2a9e
```

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, TERMINATED_HTTPS or UDP (supported only in Octavia). 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.
- `description` - (Optional) Human-readable description for the Listener.
- `connection_limit` - (Optional) The maximum number of connections allowed for the Listener.
- `timeout_client_data` - (Optional) The client inactivity timeout in milliseconds.
- `timeout_member_connect` - (Optional) The member connection timeout in milliseconds.
- `timeout_member_data` - (Optional) The member inactivity timeout in milliseconds.
- `timeout_tcp_inspect` - (Optional) The time in milliseconds, to wait for additional TCP packets for content inspection.
- `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 (https://wiki.openstack.org/wiki/Network/LBaaS/docs/how-to-create-tls-loadbalancer) for more information.

- `sni_container_refs` - (Optional) A list of references to Barbican Secrets containers which store SNI information. See here (https://wiki.openstack.org/wiki/Network/LBaaS/docs/how-to-create-tls-loadbalancer) 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.
- `timeout_client_data` - See Argument Reference above.
- `timeout_member_connect` - See Argument Reference above.
- `timeout_member_data` - See Argument Reference above.
- `timeout_tcp_inspect` - 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.

Import

Load Balancer Listener can be imported using the Listener ID, e.g.:

```
$ terraform import openstack_lb_listener_v2.listener_1 b67ce64e-8b26-405d-afeb-4a078901f15a
```

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_id` - (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_id` - 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.

Import

Load Balancer can be imported using the Load Balancer ID, e.g.:

```
$ terraform import openstack_lb_loadbalancer_v2.loadbalancer_1 19bcfdc7-c521-4a7e-9459-6750bd16df76
```

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_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.

Import

Load Balancer Pool Member can be imported using the Pool ID and Member ID separated by a slash. e.g.:

```
$ terraform import openstack_lb_member_v2.member_1 c22974d2-4c95-4bcb-9819-0afc5ed303d5/9563b79c-8460-47da-8a95-2711b746510f
```

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) 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". 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_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, HTTPS, TLS-HELLO or UDP-CONNECT (supported only in Octavia), 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.

Import

Load Balancer Pool Monitor can be imported using the Monitor ID, e.g.:

```
$ terraform import openstack_lb_monitor_v2.monitor_1 47c26fc3-2403-427a-8c79-1589bd0533c2
```

In case of using OpenContrail, the import may not work properly. If you face an issue, try to import the monitor providing its parent pool ID:

```
$ terraform import openstack_lb_monitor_v2.monitor_1 47c26fc3-2403-427a-8c79-1589bd0533c2/708bc224-0f8c-4
981-ac82-97095fe051b6
```

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_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        = "APP_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, HTTPS, PROXY or UDP (supported only in Octavia). 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.

Import

Load Balancer Pool can be imported using the Pool ID, e.g.:

```
$ terraform import openstack_lb_pool_v2.pool_1 60ad9ee4-249a-4d60-a45b-aa60e046c513
```

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_networking_addressscope_v2

Manages a V2 Neutron addressscope resource within OpenStack.

Example Usage

Create an Address-scope

```
resource "openstack_networking_addressscope_v2" "addressscope_1" {
  name       = "addressscope_1"
  ip_version = 6
}
```

Create a Subnet Pool from an Address-scope

```
resource "openstack_networking_addressscope_v2" "addressscope_1" {
  name       = "addressscope_1"
  ip_version = 6
}

resource "openstack_networking_subnetpool_v2" "subnetpool_1" {
  name             = "subnetpool_1"
  prefixes         = ["fdf7:b13d:dead:beef::/64", "fd65:86cc:a334:39b7::/64"]
  address_scope_id = "${openstack_networking_addressscope_v2.addressscope_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 address-scope. If omitted, the `region` argument of the provider is used. Changing this creates a new address-scope.
- `name` - (Required) The name of the address-scope. Changing this updates the name of the existing address-scope.
- `ip_version` - (Optional) IP version, either 4 (default) or 6. Changing this creates a new address-scope.
- `shared` - (Optional) Indicates whether this address-scope is shared across all projects. Changing this updates the shared status of the existing address-scope.
- `project_id` - (Optional) The owner of the address-scope. Required if admin wants to create a address-scope for another project. Changing this creates a new address-scope.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `ip_version` - See Argument Reference above.
- `shared` - See Argument Reference above.
- `project_id` - See Argument Reference above.

Import

Address-scopes can be imported using the `id` , e.g.

```
$ terraform import openstack_networking_addressscope_v2.addressscope_1 9cc35860-522a-4d35-974d-51d4b01180
1e
```

openstack_networking_floatingip_associate_v2

Associates a floating IP to a port. This is useful for situations where you have a pre-allocated floating IP or are unable to use the `openstack_networking_floatingip_v2` resource to create a floating IP.

Example Usage

```
resource "openstack_networking_port_v2" "port_1" {
  network_id = "a5bbd213-e1d3-49b6-aed1-9df60ea94b9a"
}

resource "openstack_networking_floatingip_associate_v2" "fip_1" {
  floating_ip = "1.2.3.4"
  port_id     = "${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 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).
- `floating_ip` - (Required) IP Address of an existing floating IP.
- `port_id` - (Required) ID of an existing port with at least one IP address to associate with this floating IP.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `floating_ip` - See Argument Reference above.
- `port_id` - See Argument Reference above.

Import

Floating IP associations can be imported using the `id` of the floating IP, e.g.

```
$ terraform import openstack_networking_floatingip_associate_v2.fip 2c7f39f3-702b-48d1-940c-b50384177ee1
```

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).
- `description` - (Optional) Human-readable description for the floating IP.
- `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)
- `address` - (Optional) The actual/specific floating IP to obtain. By default, non-admin users are not able to specify a floating IP, so you must either be an admin user or have had a custom policy or role applied to your OpenStack user or project.
- `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.
- `tags` - (Optional) A set of string tags for the floating IP.
- `dns_name` - (Optional) The floating IP DNS name. Available, when Neutron DNS extension is enabled. The data in this attribute will be published in an external DNS service when Neutron is configured to integrate with such a service. Changing this creates a new floating IP.

- `dns_domain` - (Optional) The floating IP DNS domain. Available, when Neutron DNS extension is enabled. The data in this attribute will be published in an external DNS service when Neutron is configured to integrate with such a service. Changing this creates a new floating IP.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `description` - 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.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the floating IP, which have been explicitly and implicitly added.
- `dns_name` - See Argument Reference above.
- `dns_domain` - See Argument Reference above.

Import

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

```
$ terraform import openstack_networking_floatingip_v2.floatip_1 2c7f39f3-702b-48d1-940c-b50384177ee1
```

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.
- `description` - (Optional) Human-readable description of the network. Changing this updates the name of the existing network.
- `shared` - (Optional) Specifies whether the network resource can be accessed by any tenant or not. Changing this updates the sharing capabilities of the existing network.
- `external` - (Optional) Specifies whether the network resource has the external routing facility. Valid values are true and false. Defaults to false. Changing this updates the external attribute 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.
- `tags` - (Optional) A set of string tags for the network.
- `transparent_vlan` - (Optional) Specifies whether the network resource has the VLAN transparent attribute set. Valid values are true and false. Defaults to false. Changing this updates the `transparent_vlan` attribute of the existing network.
- `port_security_enabled` - (Optional) Whether to explicitly enable or disable port security on the network. Port Security is usually enabled by default, so omitting this argument will usually result in a value of "true". Setting this explicitly to `false` will disable port security. Valid values are `true` and `false`.
- `mtu` - (Optional) The network MTU. Available for read-only, when Neutron `net-mtu` extension is enabled. Available for the modification, when Neutron `net-mtu-writable` extension is enabled.
- `dns_domain` - (Optional) The network DNS domain. Available, when Neutron DNS extension is enabled. The `dns_domain` of a network in conjunction with the `dns_name` attribute of its ports will be published in an external DNS service when Neutron is configured to integrate with such a service.
- `qos_policy_id` - (Optional) Reference to the associated QoS policy.

The `segments` block supports:

- `physical_network` - The physical 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.
- `description` - See Argument Reference above.
- `shared` - See Argument Reference above.
- `external` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `availability_zone_hints` - See Argument Reference above.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the network, which have been explicitly and implicitly added.
- `transparent_vlan` - See Argument Reference above.
- `port_security_enabled` - See Argument Reference above.
- `mtu` - See Argument Reference above.
- `dns_domain` - See Argument Reference above.
- `qos_policy_id` - 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-152ce412b6b9
```

openstack_networking_port_secgroup_associate_v2

Manages a V2 port's security groups within OpenStack. Useful, when the port was created not by Terraform (e.g. Manila or LBaaS). It should not be used, when the port was created directly within Terraform.

When the resource is deleted, Terraform doesn't delete the port, but unsets the list of user defined security group IDs. However, if `enforce` is set to `true` and the resource is deleted, Terraform will remove all assigned security group IDs.

Example Usage

Append a security group to an existing port

```
data "openstack_networking_port_v2" "system_port" {
  fixed_ip = "10.0.0.10"
}

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

resource "openstack_networking_port_secgroup_associate_v2" "port_1" {
  port_id = "${data.openstack_networking_port_v2.system_port.id}"
  security_group_ids = [
    "${data.openstack_networking_secgroup_v2.secgroup.id}",
  ]
}
```

Enforce a security group to an existing port

```
data "openstack_networking_port_v2" "system_port" {
  fixed_ip = "10.0.0.10"
}

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

resource "openstack_networking_port_secgroup_associate_v2" "port_1" {
  port_id = "${data.openstack_networking_port_v2.system_port.id}"
  enforce = "true"
  security_group_ids = [
    "${data.openstack_networking_secgroup_v2.secgroup.id}",
  ]
}
```

Remove all security groups from an existing port

```
data "openstack_networking_port_v2" "system_port" {
  fixed_ip = "10.0.0.10"
}

resource "openstack_networking_port_secgroup_associate_v2" "port_1" {
  port_id            = "${data.openstack_networking_port_v2.system_port.id}"
  enforce            = "true"
  security_group_ids = []
}
```

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 manage a port. If omitted, the `region` argument of the provider is used. Changing this creates a new resource.
- `port_id` - (Required) An UUID of the port to apply security groups to.
- `security_group_ids` - (Required) 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).
- `enforce` - (Optional) Whether to replace or append the list of security groups, specified in the `security_group_ids`. Defaults to `false`.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `port_id` - See Argument Reference above.
- `security_group_ids` - See Argument Reference above.
- `all_security_group_ids` - The collection of Security Group IDs on the port which have been explicitly and implicitly added.

openstack_networking_port_v2

Manages a V2 port resource within OpenStack.

Example Usage

Simple port

```
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"
}
```

Port with physical binding information

```
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}"
  device_id      = "cdf70fcf-c161-4f24-9c70-96b3f5a54b71"
  device_owner   = "baremetal:none"
  admin_state_up = "true"

  binding = {
    host_id   = "b080b9cf-46e0-4ce8-ad47-0fd4accc872b"
    vnic_type = "baremetal"
    profile   = <<EOF
{
  "local_link_information": [
    {
      "switch_info": "info1",
      "port_id": "Ethernet3/4",
      "switch_id": "12:34:56:78:9A:BC"
    },
    {
      "switch_info": "info2",
      "port_id": "Ethernet3/4",
      "switch_id": "12:34:56:78:9A:BD"
    }
  ],
  "vlan_type": "allowed"
}
EOF
  }
}
```

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.
- `description` - (Optional) Human-readable description of the floating IP. Changing this updates the `description` 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 - Conflicts with `no_fixed_ip`) An array of desired IPs for this port. The structure is described below.
- `no_fixed_ip` - (Optional - Conflicts with `fixed_ip`) Create a port with no fixed IP address. This will also remove any fixed IPs previously set on a port. `true` is the only valid value for this argument.
- `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.
- `extra_dhcp_option` - (Optional) An extra DHCP option that needs to be configured on the port. The structure is described below. Can be specified multiple times.
- `port_security_enabled` - (Optional) Whether to explicitly enable or disable port security on the port. Port Security is usually enabled by default, so omitting argument will usually result in a value of "true". Setting this explicitly to `false` will disable port security. In order to disable port security, the port must not have any security groups. Valid values are `true` and `false`.
- `value_specs` - (Optional) Map of additional options.
- `tags` - (Optional) A set of string tags for the port.
- `binding` - (Optional) The port binding allows to specify binding information for the port. The structure is described below.
- `dns_name` - (Optional) The port DNS name. Available, when Neutron DNS extension is enabled.
- `qos_policy_id` - (Optional) Reference to the associated QoS policy.

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 or omitted. 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.

The `extra_dhcp_option` block supports:

- `name` - (Required) Name of the DHCP option.
- `value` - (Required) Value of the DHCP option.
- `ip_version` - (Optional) IP protocol version. Defaults to 4.

The `binding` block supports:

- `host_id` - (Optional) The ID of the host to allocate port on.
- `profile` - (Optional) Custom data to be passed as `binding:profile`. Data must be passed as JSON.
- `vnic_type` - (Optional) VNIC type for the port. Can either be `direct`, `direct-physical`, `macvtap`, `normal`, `baremetal` or `virtio-forwarder`. Default value is `normal`.
- `vif_details` - (Computed) A map of JSON strings containing additional details for this specific binding.
- `vif_type` - (Computed) The VNIC type of the port binding.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `description` - 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.
- `extra_dhcp_option` - See Argument Reference above.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the port, which have been explicitly and implicitly added.
- `binding` - See Argument Reference above.
- `dns_name` - See Argument Reference above.

- `dns_assignment` - The list of maps representing port DNS assignments.
- `qos_policy_id` - See Argument Reference above.

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_qos_bandwidth_limit_rule_v2

Manages a V2 Neutron QoS bandwidth limit rule resource within OpenStack.

Example Usage

Create a QoS Policy with some bandwidth limit rule

```
resource "openstack_networking_qos_policy_v2" "qos_policy_1" {
  name        = "qos_policy_1"
  description = "bw_limit"
}

resource "openstack_networking_qos_bandwidth_limit_rule_v2" "bw_limit_rule_1" {
  qos_policy_id  = "${openstack_networking_qos_policy_v2.qos_policy_1.id}"
  max_kbps       = 3000
  max_burst_kbps = 300
  direction      = "egress"
}
```

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 QoS bandwidth limit rule. If omitted, the `region` argument of the provider is used. Changing this creates a new QoS bandwidth limit rule.
- `qos_policy_id` - (Required) The QoS policy reference. Changing this creates a new QoS bandwidth limit rule.
- `max_kbps` - (Required) The maximum kilobits per second of a QoS bandwidth limit rule. Changing this updates the maximum kilobits per second of the existing QoS bandwidth limit rule.
- `max_burst_kbps` - (Optional) The maximum burst size in kilobits of a QoS bandwidth limit rule. Changing this updates the maximum burst size in kilobits of the existing QoS bandwidth limit rule.
- `direction` - (Optional) The direction of traffic. Defaults to "egress". Changing this updates the direction of the existing QoS bandwidth limit rule.

Attributes Reference

The following attributes are exported:

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

- `max_kbps` - See Argument Reference above.
- `max_burst_kbps` - See Argument Reference above.
- `direction` - See Argument Reference above.

Import

QoS bandwidth limit rules can be imported using the `qos_policy_id/bandwidth_limit_rule` format, e.g.

```
$ terraform import openstack_networking_qos_bandwidth_limit_rule_v2.bw_limit_rule_1 d6ae28ce-fcb5-4180-aa
62-d260a27e09ae/46dfb556-b92f-48ce-94c5-9a9e2140de94
```

openstack_networking_qos_dscp_marking_rule_v2

Manages a V2 Neutron QoS DSCP marking rule resource within OpenStack.

Example Usage

Create a QoS Policy with some DSCP marking rule

```
resource "openstack_networking_qos_policy_v2" "qos_policy_1" {
  name        = "qos_policy_1"
  description = "dscp_mark"
}

resource "openstack_networking_qos_dscp_marking_rule_v2" "dscp_marking_rule_1" {
  qos_policy_id = "${openstack_networking_qos_policy_v2.qos_policy_1.id}"
  dscp_mark     = 26
}
```

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 QoS DSCP marking rule. If omitted, the `region` argument of the provider is used. Changing this creates a new QoS DSCP marking rule.
- `qos_policy_id` - (Required) The QoS policy reference. Changing this creates a new QoS DSCP marking rule.
- `dscp_mark` - (Required) The value of DSCP mark. Changing this updates the DSCP mark value existing QoS DSCP marking rule.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `qos_policy_id` - See Argument Reference above.
- `dscp_mark` - See Argument Reference above.

Import

QoS DSCP marking rules can be imported using the `qos_policy_id/dscp_marking_rule_id` format, e.g.

```
$ terraform import openstack_networking_qos_dscp_marking_rule_v2.dscp_marking_rule_1 d6ae28ce-fcb5-4180-a
a62-d260a27e09ae/46dfb556-b92f-48ce-94c5-9a9e2140de94
```

openstack_networking_qos_minimum_bandwidth_rule_v2

Manages a V2 Neutron QoS minimum bandwidth rule resource within OpenStack.

Example Usage

Create a QoS Policy with some minimum bandwidth rule

```
resource "openstack_networking_qos_policy_v2" "qos_policy_1" {
  name        = "qos_policy_1"
  description = "min_kbps"
}

resource "openstack_networking_qos_minimum_bandwidth_rule_v2" "minimum_bandwidth_rule_1" {
  qos_policy_id = "${openstack_networking_qos_policy_v2.qos_policy_1.id}"
  min_kbps      = 200
}
```

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 QoS minimum bandwidth rule. If omitted, the `region` argument of the provider is used. Changing this creates a new QoS minimum bandwidth rule.
- `qos_policy_id` - (Required) The QoS policy reference. Changing this creates a new QoS minimum bandwidth rule.
- `min_kbps` - (Required) The minimum kilobits per second. Changing this updates the min kbps value of the existing QoS minimum bandwidth rule.
- `direction` - (Optional) The direction of traffic. Defaults to "egress". Changing this updates the direction of the existing QoS minimum bandwidth rule.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `qos_policy_id` - See Argument Reference above.
- `min_kbps` - See Argument Reference above.
- `direction` - See Argument Reference above.

Import

QoS minimum bandwidth rules can be imported using the `qos_policy_id/minimum_bandwidth_rule_id` format, e.g.

```
$ terraform import openstack_networking_qos_minimum_bandwidth_rule_v2.minimum_bandwidth_rule_1 d6ae28ce-f
cb5-4180-aa62-d260a27e09ae/46dfb556-b92f-48ce-94c5-9a9e2140de94
```

openstack_networking_qos_policy_v2

Manages a V2 Neutron QoS policy resource within OpenStack.

Example Usage

Create a QoS Policy

```
resource "openstack_networking_qos_policy_v2" "qos_policy_1" {
  name        = "qos_policy_1"
  description = "bw_limit"
}
```

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 Qos policy. If omitted, the `region` argument of the provider is used. Changing this creates a new QoS policy.
- `name` - (Required) The name of the QoS policy. Changing this updates the name of the existing QoS policy.
- `project_id` - (Optional) The owner of the QoS policy. Required if admin wants to create a QoS policy for another project. Changing this creates a new QoS policy.
- `shared` - (Optional) Indicates whether this QoS policy is shared across all projects. Changing this updates the shared status of the existing QoS policy.
- `description` - (Optional) The human-readable description for the QoS policy. Changing this updates the description of the existing QoS policy.
- `is_default` - (Optional) Indicates whether the QoS policy is default QoS policy or not. Changing this updates the default status of the existing QoS policy.
- `value_specs` - (Optional) Map of additional options.
- `tags` - (Optional) A set of string tags for the QoS policy.

Attributes Reference

The following attributes are exported:

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

- `created_at` - The time at which QoS policy was created.
- `updated_at` - The time at which QoS policy was created.
- `shared` - See Argument Reference above.
- `description` - See Argument Reference above.
- `is_default` - See Argument Reference above.
- `revision_number` - The revision number of the QoS policy.
- `value_specs` - See Argument Reference above.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the QoS policy, which have been explicitly and implicitly added.

Import

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

```
$ terraform import openstack_networking_qos_policy_v2.qos_policy_1 d6ae28ce-fcb5-4180-aa62-d260a27e09ae
```

openstack_networking_quota_v2

Manages a V2 networking quota resource within OpenStack.

Note: This usually requires admin privileges.

Note: This resource has a no-op deletion so no actual actions will be done against the OpenStack API in case of delete call.

Example Usage

```
resource "openstack_identity_project_v2" "project_1" {
  name = project_1
}

resource "openstack_networking_quota_v2" "quota_1" {
  project_id          = "${openstack_identity_project_v2.project_1.id}"
  floatingip          = 10
  network             = 4
  port                = 100
  rbac_policy         = 10
  router              = 4
  security_group      = 10
  security_group_rule = 100
  subnet              = 8
  subnetpool          = 2
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to create the quota. If omitted, the `region` argument of the provider is used. Changing this creates new quota.
- `project_id` - (Required) ID of the project to manage quota. Changing this creates new quota.
- `floatingip` - (Optional) Quota value for floating IPs. Changing this updates the existing quota.
- `network` - (Optional) Quota value for networks. Changing this updates the existing quota.
- `port` - (Optional) Quota value for ports. Changing this updates the existing quota.
- `rbac_policy` - (Optional) Quota value for RBAC policies. Changing this updates the existing quota.
- `router` - (Optional) Quota value for routers. Changing this updates the existing quota.
- `security_group` - (Optional) Quota value for security groups. Changing this updates the existing quota.

- `security_group_rule` - (Optional) Quota value for security group rules. Changing this updates the existing quota.
- `subnet` - (Optional) Quota value for subnets. Changing this updates the existing quota.
- `subnetpool` - (Optional) Quota value for subnetpools. Changing this updates the existing quota.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `project_id` - See Argument Reference above.
- `floatingip` - See Argument Reference above.
- `network` - See Argument Reference above.
- `port` - See Argument Reference above.
- `rbac_policy` - See Argument Reference above.
- `router` - See Argument Reference above.
- `security_group` - See Argument Reference above.
- `security_group_rule` - See Argument Reference above.
- `subnet` - See Argument Reference above.
- `subnetpool` - See Argument Reference above.

Import

Quotas can be imported using the `project_id`, e.g.

```
$ terraform import openstack_networking_quota_v2.quota_1 2a0f2240-c5e6-41de-896d-e80d97428d6b
```

openstack_networking_rbac_policy_v2

The RBAC policy resource contains functionality for working with Neutron RBAC Policies. Role-Based Access Control (RBAC) policy framework enables both operators and users to grant access to resources for specific projects.

Sharing an object with a specific project is accomplished by creating a policy entry that permits the target project the `access_as_shared` action on that object.

To make a network available as an external network for specific projects rather than all projects, use the `access_as_external` action. If a network is marked as external during creation, it now implicitly creates a wildcard RBAC policy granting everyone access to preserve previous behavior before this feature was added.

Example Usage

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

resource "openstack_networking_rbac_policy_v2" "rbac_policy_1" {
  action        = "access_as_shared"
  object_id     = "${openstack_networking_network_v2.network_1.id}"
  object_type   = "network"
  target_tenant = "20415a973c9e45d3917f078950644697"
}
```

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 subnet. If omitted, the `region` argument of the provider is used. Changing this creates a new routing entry.
- `action` - (Required) Action for the RBAC policy. Can either be `access_as_external` or `access_as_shared`.
- `object_id` - (Required) The ID of the `object_type` resource. An `object_type` of `network` returns a network ID and an `object_type` of `qos_policy` returns a QoS ID.
- `object_type` - (Required) The type of the object that the RBAC policy affects. Can either be `qos-policy` or `network`.
- `target_tenant` - (Required) The ID of the tenant to which the RBAC policy will be enforced.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `action` - See Argument Reference above.
- `object_id` - See Argument Reference above.
- `object_type` - See Argument Reference above.
- `target_tenant` - See Argument Reference above.
- `tenant_id` - The owner of the RBAC policy.

Notes

Import

RBAC policies can be imported using the `id`, e.g.

```
$ terraform import openstack_networking_rbac_policy_v2.rbac_policy_1 eae26a3e-1c33-4cc1-9c31-0cd729c438a1
```

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_network_id = "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-4f70-9f6c-281607dad0
79-route-10.0.1.0/24-192.168.199.25
```

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.
- `description` - (Optional) Human-readable description for the 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.

- `tags` - (Optional) A set of string tags for the router.
- `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.
- `description` - 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.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the router, which have been explicitly and implicitly added.

Import

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

```
$ terraform import openstack_networking_router_v2.router_1 014395cd-89fc-4c9b-96b7-13d1ee79dad2
```

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.
- `description` - (Optional) A description of the rule. 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.
- `description` - 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-4256-955c-9584a6212745
```

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.
- `tags` - (Optional) A set of string tags for the security group.

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.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the security group, which have been explicitly and implicitly added.

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-4852-9139-6f461c90e8bc
```

openstack_networking_subnetpool_v2

Manages a V2 Neutron subnetpool resource within OpenStack.

Example Usage

Create a Subnet Pool

```
resource "openstack_networking_subnetpool_v2" "subnetpool_1" {
  name       = "subnetpool_1"
  ip_version = 6
  prefixes   = ["fdf7:b13d:dead:beef::/64", "fd65:86cc:a334:39b7::/64"]
}
```

Create a Subnet from a Subnet Pool

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

resource "openstack_networking_subnetpool_v2" "subnetpool_1" {
  name     = "subnetpool_1"
  prefixes = ["10.11.12.0/24"]
}

resource "openstack_networking_subnet_v2" "subnet_1" {
  name          = "subnet_1"
  cidr          = "10.11.12.0/25"
  network_id    = "${openstack_networking_network_v2.network_1.id}"
  subnetpool_id = "${openstack_networking_subnetpool_v2.subnetpool_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 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.
- `tags` - (Optional) A set of string tags for the subnetpool.

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.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the subnetpool, which have been explicitly and implicitly added.

Import

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

```
$ terraform import openstack_networking_subnetpool_v2.subnetpool_1 832cb7f3-59fe-40cf-8f64-8350ffc03272
```

openstack_networking_subnet_route_v2

Creates a routing entry on a OpenStack V2 subnet.

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_subnet_route_v2" "subnet_route_1" {
  subnet_id        = "${openstack_networking_subnet_v2.subnet_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 subnet. If omitted, the `region` argument of the provider is used. Changing this creates a new routing entry.
- `subnet_id` - (Required) ID of the subnet 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.

- `subnet_id` - See Argument Reference above.
- `destination_cidr` - See Argument Reference above.
- `next_hop` - See Argument Reference above.

Notes

Import

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

```
$ terraform import openstack_networking_subnet_route_v2.subnet_route_1 686fe248-386c-4f70-9f6c-281607dad0
79-route-10.0.1.0/24-192.168.199.25
```

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` - (Optional) CIDR representing IP range for this subnet, based on IP version. You can omit this option if you are creating a subnet from a subnet pool.
- `prefix_length` - (Optional) The prefix length to use when creating a subnet from a subnet pool. The default subnet pool prefix length that was defined when creating the subnet pool will be used if not provided. 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.
- `description` - (Optional) Human-readable description 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` - (**Deprecated** - use `allocation_pool` instead) A block declaring the start and end range of the IP addresses available for use with DHCP in this subnet. The `allocation_pools` block is documented below.

- `allocation_pool` - (Optional) A block declaring the start and end range of the IP addresses available for use with DHCP in this subnet. Multiple `allocation_pool` blocks can be declared, providing the subnet with more than one range of IP addresses to use with DHCP. However, each IP range must be from the same CIDR that the subnet is part of. The `allocation_pool` block is documented below.
- `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` - (**Deprecated** - use `openstack_networking_subnet_route_v2` instead) 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.
- `tags` - (Optional) A set of string tags for the subnet.

The deprecated `allocation_pools` block supports:

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

The `allocation_pool` 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.
- `description` - 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.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of ags assigned on the subnet, which have been explicitly and implicitly added.

Import

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

```
$ terraform import openstack_networking_subnet_v2.subnet_1 da4faf16-5546-41e4-8330-4d0002b74048
```

openstack_networking_trunk_v2

Manages a networking V2 trunk 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        = "192.168.1.0/24"
  ip_version  = 4
  enable_dhcp = true
  no_gateway  = true
}

resource "openstack_networking_port_v2" "parent_port_1" {
  depends_on = [
    "openstack_networking_subnet_v2.subnet_1",
  ]

  name           = "parent_port_1"
  network_id     = "${openstack_networking_network_v2.network_1.id}"
  admin_state_up = "true"
}

resource "openstack_networking_port_v2" "subport_1" {
  depends_on = [
    "openstack_networking_subnet_v2.subnet_1",
  ]

  name           = "subport_1"
  network_id     = "${openstack_networking_network_v2.network_1.id}"
  admin_state_up = "true"
}

resource "openstack_networking_trunk_v2" "trunk_1" {
  name           = "trunk_1"
  admin_state_up = "true"
  port_id        = "${openstack_networking_port_v2.parent_port_1.id}"

  sub_port {
    port_id           = "${openstack_networking_port_v2.subport_1.id}"
    segmentation_id   = 1
    segmentation_type = "vlan"
  }
}

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

  network {
    port = "${openstack_networking_trunk_v2.trunk_1.port_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 trunk. If omitted, the `region` argument of the provider is used. Changing this creates a new trunk.
- `name` - (Optional) A unique name for the trunk. Changing this updates the `name` of an existing trunk.
- `description` - (Optional) Human-readable description of the trunk. Changing this updates the name of the existing trunk.
- `port_id` - (Required) The ID of the port to be used as the parent port of the trunk. This is the port that should be used as the compute instance network port. Changing this creates a new trunk.
- `admin_state_up` - (Optional) Administrative up/down status for the trunk (must be "true" or "false" if provided). Changing this updates the `admin_state_up` of an existing trunk.
- `tenant_id` - (Optional) The owner of the Trunk. Required if admin wants to create a trunk on behalf of another tenant. Changing this creates a new trunk.
- `sub_port` - (Optional) The set of ports that will be made subports of the trunk. The structure of each subport is described below.
- `tags` - (Optional) A set of string tags for the port.

The `sub_port` block supports:

- `port_id` - (Required) The ID of the port to be made a subport of the trunk.
- `segmentation_type` - (Required) The segmentation technology to use, e.g., "vlan".
- `segmentation_id` - (Required) The numeric id of the subport segment.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `port_id` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `sub_port` - See Argument Reference above.
- `tags` - See Argument Reference above.
- `all_tags` - The collection of tags assigned on the trunk, which have been explicitly and implicitly added.

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"

  versioning {
    type     = "versions"
    location = "tf-test-container-versions"
  }
}
```

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.
- `versioning` - (Optional) Enable object versioning. The structure is described below.
- `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.

The `versioning` block supports:

- `type` - (Required) Versioning type which can be `versions` or `history` according to Openstack documentation (https://docs.openstack.org/swift/latest/overview_object_versioning.html).
- `location` - (Required) Container in which versions will be stored.

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.
- `versioning` - 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.

openstack_objectstorage_tempurl_v1

Use this resource to generate an OpenStack Object Storage temporary URL.

The temporary URL will be valid for as long as TTL is set to (in seconds). Once the URL has expired, it will no longer be valid, but the resource will remain in place. If you wish to automatically regenerate a URL, set the `regenerate` argument to `true`. This will create a new resource with a new ID and URL.

Example Usage

```
resource "openstack_objectstorage_tempurl_v1" "obj_tempurl" {
  container = "test"
  object    = "container"
  method    = "post"
  ttl       = 20
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region the tempurl is located in.
- `container` - (Required) The container name the object belongs to.
- `object` - (Required) The object name the tempurl is for.
- `ttl` - (Required) The TTL, in seconds, for the URL. For how long it should be valid.
- `method` - (Optional) The method allowed when accessing this URL. Valid values are `GET`, and `POST`. Default is `GET`.
- `regenerate` - (Optional) Whether to automatically regenerate the URL when it has expired. If set to true, this will create a new resource with a new ID and new URL. Defaults to false.

Attributes Reference

- `id` - Computed md5 hash based on the generated url
- `container` - See Argument Reference above.
- `object` - See Argument Reference above.
- `ttl` - See Argument Reference above.
- `method` - See Argument Reference above.
- `url` - The URL
- `region` - The region the endpoint is located in.

sharedfilesystem_securityservice_v2

Use this resource to configure a security service.

A security service stores configuration information for clients for authentication and authorization (AuthN/AuthZ). For example, a share server will be the client for an existing service such as LDAP, Kerberos, or Microsoft Active Directory.

Minimum supported Manila microversion is 2.7.

Example Usage

```
resource "openstack_sharedfilesystem_securityservice_v2" "securityservice_1" {
  name        = "security"
  description = "created by terraform"
  type        = "active_directory"
  server      = "192.168.199.10"
  dns_ip      = "192.168.199.10"
  domain      = "example.com"
  ou          = "CN=Computers,DC=example,DC=com"
  user        = "joinDomainUser"
  password    = "s8cret"
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Shared File System client. A Shared File System client is needed to create a security service. If omitted, the `region` argument of the provider is used. Changing this creates a new security service.
- `name` - (Optional) The name of the security service. Changing this updates the name of the existing security service.
- `description` - (Optional) The human-readable description for the security service. Changing this updates the description of the existing security service.
- `type` - (Required) The security service type - can either be active_directory, kerberos or ldap. Changing this updates the existing security service.
- `dns_ip` - (Optional) The security service DNS IP address that is used inside the tenant network.
- `ou` - (Optional) The security service ou. An organizational unit can be added to specify where the share ends up. New in Manila microversion 2.44.
- `user` - (Optional) The security service user or group name that is used by the tenant.
- `password` - (Optional) The user password, if you specify a user.
- `domain` - (Optional) The security service domain.
- `server` - (Optional) The security service host name or IP address.

Attributes Reference

- `id` - The unique ID for the Security Service.
- `region` - See Argument Reference above.
- `project_id` - The owner of the Security Service.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `type` - See Argument Reference above.
- `dns_ip` - See Argument Reference above.
- `ou` - See Argument Reference above.
- `user` - See Argument Reference above.
- `password` - See Argument Reference above.
- `domain` - See Argument Reference above.
- `server` - See Argument Reference above.

Import

This resource can be imported by specifying the ID of the security service:

```
$ terraform import openstack_sharedfilesystem_securityservice_v2.securityservice_1 <id>
```

openstack_sharedfilesystem_share_access_v2

Use this resource to control the share access lists.

Important Security Notice The access key retrieved by this resource will be stored *unencrypted* in your Terraform state file. If you use this resource in production, please make sure your state file is sufficiently protected.

Example Usage

NFS

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

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

resource "openstack_sharedfilesystem_sharenetwork_v2" "sharenetwork_1" {
  name              = "test_sharenetwork"
  description       = "test share network with security services"
  neutron_net_id    = "${openstack_networking_network_v2.network_1.id}"
  neutron_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}

resource "openstack_sharedfilesystem_share_v2" "share_1" {
  name             = "nfs_share"
  description      = "test share description"
  share_proto      = "NFS"
  size             = 1
  share_network_id = "${openstack_sharedfilesystem_sharenetwork_v2.sharenetwork_1.id}"
}

resource "openstack_sharedfilesystem_share_access_v2" "share_access_1" {
  share_id     = "${openstack_sharedfilesystem_share_v2.share_1.id}"
  access_type  = "ip"
  access_to    = "192.168.199.10"
  access_level = "rw"
}
```

CIFS

```
resource "openstack_networking_network_v2" "network_1" {
```

```
  name           = "network_1"
  admin_state_up = "true"
}

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

resource "openstack_sharedfilesystem_securityservice_v2" "securityservice_1" {
  name        = "security"
  description = "created by terraform"
  type        = "active_directory"
  server      = "192.168.199.10"
  dns_ip      = "192.168.199.10"
  domain      = "example.com"
  ou          = "CN=Computers,DC=example,DC=com"
  user        = "joinDomainUser"
  password    = "s8cret"
}

resource "openstack_sharedfilesystem_sharenetwork_v2" "sharenetwork_1" {
  name              = "test_sharenetwork_secure"
  description       = "share the secure love"
  neutron_net_id    = "${openstack_networking_network_v2.network_1.id}"
  neutron_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
  security_service_ids = [
    "${openstack_sharedfilesystem_securityservice_v2.securityservice_1.id}",
  ]
}

resource "openstack_sharedfilesystem_share_v2" "share_1" {
  name             = "cifs_share"
  share_proto      = "CIFS"
  size             = 1
  share_network_id = "${openstack_sharedfilesystem_sharenetwork_v2.sharenetwork_1.id}"
}

resource "openstack_sharedfilesystem_share_access_v2" "share_access_1" {
  share_id     = "${openstack_sharedfilesystem_share_v2.share_1.id}"
  access_type  = "user"
  access_to    = "windows"
  access_level = "ro"
}

resource "openstack_sharedfilesystem_share_access_v2" "share_access_2" {
  share_id     = "${openstack_sharedfilesystem_share_v2.share_1.id}"
  access_type  = "user"
  access_to    = "linux"
  access_level = "rw"
}

output "export_locations" {
  value = "${openstack_sharedfilesystem_share_v2.share_1.export_locations}"
}
```

Argument Reference

The following arguments are supported:

- `region` - The region in which to obtain the V2 Shared File System client. A Shared File System client is needed to create a share access. Changing this creates a new share access.
- `share_id` - (Required) The UUID of the share to which you are granted access.
- `access_type` - (Required) The access rule type. Can either be an ip, user, cert, or cephx. cephx support requires an OpenStack environment that supports Shared Filesystem microversion 2.13 (Mitaka) or later.
- `access_to` - (Required) The value that defines the access. Can either be an IP address or a username verified by configured Security Service of the Share Network.
- `access_level` - (Required) The access level to the share. Can either be `rw` or `ro`.

Attributes Reference

- `id` - The unique ID for the Share Access.
- `region` - See Argument Reference above.
- `share_id` - See Argument Reference above.
- `access_type` - See Argument Reference above.
- `access_to` - See Argument Reference above.
- `access_level` - See Argument Reference above.
- `access_key` - The access credential of the entity granted access.

Import

This resource can be imported by specifying the ID of the share and the ID of the share access, separated by a slash, e.g.:

```
$ terraform import openstack_sharedfilesystem_share_access_v2.share_access_1 <share id>/<share access id>
```

sharedfilesystem_sharenetwork_v2

Use this resource to configure a share network.

A share network stores network information that share servers can use when shares are created.

Example Usage

Basic share network

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

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

resource "openstack_sharedfilesystem_sharenetwork_v2" "sharenetwork_1" {
  name              = "test_sharenetwork"
  description       = "test share network"
  neutron_net_id    = "${openstack_networking_network_v2.network_1.id}"
  neutron_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}
```

Share network with associated security services

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

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

resource "openstack_sharedfilesystem_securityservice_v2" "securityservice_1" {
  name        = "security"
  description = "created by terraform"
  type        = "active_directory"
  server      = "192.168.199.10"
  dns_ip      = "192.168.199.10"
  domain      = "example.com"
  ou          = "CN=Computers,DC=example,DC=com"
  user        = "joinDomainUser"
  password    = "s8cret"
}

resource "openstack_sharedfilesystem_sharenetwork_v2" "sharenetwork_1" {
  name              = "test_sharenetwork"
  description       = "test share network with security services"
  neutron_net_id    = "${openstack_networking_network_v2.network_1.id}"
  neutron_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
  security_service_ids = [
    "${openstack_sharedfilesystem_securityservice_v2.securityservice_1.id}",
  ]
}
```

Argument Reference

The following arguments are supported:

- `region` - (Optional) The region in which to obtain the V2 Shared File System client. A Shared File System client is needed to create a share network. If omitted, the `region` argument of the provider is used. Changing this creates a new share network.
- `name` - (Optional) The name for the share network. Changing this updates the name of the existing share network.
- `description` - (Optional) The human-readable description for the share network. Changing this updates the description of the existing share network.
- `neutron_net_id` - (Required) The UUID of a neutron network when setting up or updating a share network. Changing this updates the existing share network if it's not used by shares.
- `neutron_subnet_id` - (Required) The UUID of the neutron subnet when setting up or updating a share network. Changing this updates the existing share network if it's not used by shares.

- `security_service_ids` - (Optional) The list of security service IDs to associate with the share network. The security service must be specified by ID and not name.

Attributes Reference

- `id` - The unique ID for the Share Network.
- `region` - See Argument Reference above.
- `project_id` - The owner of the Share Network.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `neutron_net_id` - See Argument Reference above.
- `neutron_subnet_id` - See Argument Reference above.
- `security_service_ids` - See Argument Reference above.
- `network_type` - The share network type. Can either be VLAN, VXLAN, GRE, or flat.
- `segmentation_id` - The share network segmentation ID.
- `cidr` - The share network CIDR.
- `ip_version` - The IP version of the share network. Can either be 4 or 6.

Import

This resource can be imported by specifying the ID of the share network:

```
$ terraform import openstack_sharedfilesystem_sharenetwork_v2.sharenetwork_1 <id>
```

openstack_sharedfilesystem_share_v2

Use this resource to configure a share.

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

resource "openstack_sharedfilesystem_sharenetwork_v2" "sharenetwork_1" {
  name              = "test_sharenetwork"
  description       = "test share network with security services"
  neutron_net_id    = "${openstack_networking_network_v2.network_1.id}"
  neutron_subnet_id = "${openstack_networking_subnet_v2.subnet_1.id}"
}

resource "openstack_sharedfilesystem_share_v2" "share_1" {
  name             = "nfs_share"
  description      = "test share description"
  share_proto      = "NFS"
  size             = 1
  share_network_id = "${openstack_sharedfilesystem_sharenetwork_v2.sharenetwork_1.id}"
}
```

Argument Reference

The following arguments are supported:

- `region` - The region in which to obtain the V2 Shared File System client. A Shared File System client is needed to create a share. Changing this creates a new share.
- `name` - (Optional) The name of the share. Changing this updates the name of the existing share.
- `description` - (Optional) The human-readable description for the share. Changing this updates the description of the existing share.
- `share_proto` - (Required) The share protocol - can either be NFS, CIFS, CEPHFS, GLUSTERFS, HDFS or MAPRFS. Changing this creates a new share.
- `size` - (Required) The share size, in GBs. The requested share size cannot be greater than the allowed GB quota. Changing this resizes the existing share.

- `share_type` - (Optional) The share type name. If you omit this parameter, the default share type is used.
- `snapshot_id` - (Optional) The UUID of the share's base snapshot. Changing this creates a new share.
- `is_public` - (Optional) The level of visibility for the share. Set to true to make share public. Set to false to make it private. Default value is false. Changing this updates the existing share.
- `metadata` - (Optional) One or more metadata key and value pairs as a dictionary of strings.
- `share_network_id` - (Optional) The UUID of a share network where the share server exists or will be created. If `share_network_id` is not set and you provide a `snapshot_id`, the share_network_id value from the snapshot is used. Changing this creates a new share.
- `availability_zone` - (Optional) The share availability zone. Changing this creates a new share.

Attributes Reference

- `id` - The unique ID for the Share.
- `region` - See Argument Reference above.
- `project_id` - The owner of the Share.
- `name` - See Argument Reference above.
- `description` - See Argument Reference above.
- `share_proto` - See Argument Reference above.
- `size` - See Argument Reference above.
- `share_type` - See Argument Reference above.
- `snapshot_id` - See Argument Reference above.
- `is_public` - See Argument Reference above.
- `metadata` - See Argument Reference above.
- `share_network_id` - See Argument Reference above.
- `availability_zone` - See Argument Reference above.
- `export_locations` - A list of export locations. For example, when a share server has more than one network interface, it can have multiple export locations.
- `has_replicas` - Indicates whether a share has replicas or not.
- `host` - The share host name.
- `replication_type` - The share replication type.
- `share_server_id` - The UUID of the share server.
- `all_metadata` - The map of metadata, assigned on the share, which has been explicitly and implicitly added.

Import

This resource can be imported by specifying the ID of the share:

```
$ terraform import openstack_sharedfilesystem_share_v2.share_1 <id>
```

openstack_vpnaas_endpoint_group_v2

Manages a V2 Neutron Endpoint Group resource within OpenStack.

Example Usage

```
resource "openstack_vpnaas_endpoint_group_v2" "group_1" {
  name = "Group 1"
  type = "cidr"
  endpoints = ["10.2.0.0/24",
  "10.3.0.0/24", ]
}
```

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 an endpoint group. If omitted, the `region` argument of the provider is used. Changing this creates a new group.
- `name` - (Optional) The name of the group. Changing this updates the name of the existing group.
- `tenant_id` - (Optional) The owner of the group. Required if admin wants to create an endpoint group for another project. Changing this creates a new group.
- `description` - (Optional) The human-readable description for the group. Changing this updates the description of the existing group.
- `type` - The type of the endpoints in the group. A valid value is subnet, cidr, network, router, or vlan. Changing this creates a new group.
- `endpoints` - List of endpoints of the same type, for the endpoint group. The values will depend on the type. Changing this creates a new group.
- `value_specs` - (Optional) Map of additional options.

Attributes Reference

The following attributes are exported:

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

- `endpoints` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

Import

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

```
$ terraform import openstack_vpnaas_endpoint_group_v2.group_1 832cb7f3-59fe-40cf-8f64-8350ffc03272
```

openstack_vpnaas_ike_policy_v2

Manages a V2 Neutron IKE policy resource within OpenStack.

Example Usage

```
resource "openstack_vpnaas_ike_policy_v2" "policy_1" {
  name = "my_policy"
}
```

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 VPN service. If omitted, the `region` argument of the provider is used. Changing this creates a new service.
- `name` - (Optional) The name of the policy. Changing this updates the name of the existing policy.
- `tenant_id` - (Optional) The owner of the policy. Required if admin wants to create a service for another policy. Changing this creates a new policy.
- `description` - (Optional) The human-readable description for the policy. Changing this updates the description of the existing policy.
- `auth_algorithm` - (Optional) The authentication hash algorithm. Valid values are sha1, sha256, sha384, sha512. Default is sha1. Changing this updates the algorithm of the existing policy.
- `encryption_algorithm` - (Optional) The encryption algorithm. Valid values are 3des, aes-128, aes-192 and so on. The default value is aes-128. Changing this updates the existing policy.
- `pfs` - (Optional) The perfect forward secrecy mode. Valid values are Group2, Group5 and Group14. Default is Group5. Changing this updates the existing policy.
- `phase1_negotiation_mode` - (Optional) The IKE mode. A valid value is main, which is the default. Changing this updates the existing policy.
- `ike_version` - (Optional) The IKE mode. A valid value is v1 or v2. Default is v1. Changing this updates the existing policy.
- `lifetime` - (Optional) The lifetime of the security association. Consists of Unit and Value.
  - `unit` - (Optional) The units for the lifetime of the security association. Can be either seconds or kilobytes. Default is seconds.
  - `value` - (Optional) The value for the lifetime of the security association. Must be a positive integer. Default is 3600.
- `value_specs` - (Optional) Map of additional options.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `description` - See Argument Reference above.
- `auth_algorithm` - See Argument Reference above.
- `encapsulation_mode` - See Argument Reference above.
- `encryption_algorithm` - See Argument Reference above.
- `pfs` - See Argument Reference above.
- `transform_protocol` - See Argument Reference above.
- `lifetime` - See Argument Reference above.
  - `unit` - See Argument Reference above.
  - `value` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

Import

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

```
$ terraform import openstack_vpnaas_ike_policy_v2.policy_1 832cb7f3-59fe-40cf-8f64-8350ffc03272
```

openstack_vpnaas_ipsec_policy_v2

Manages a V2 Neutron IPSec policy resource within OpenStack.

Example Usage

```
resource "openstack_vpnaas_ipsec_policy_v2" "policy_1" {
  name = "my_policy"
}
```

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 IPSec policy. If omitted, the `region` argument of the provider is used. Changing this creates a new policy.
- `name` - (Optional) The name of the policy. Changing this updates the name of the existing policy.
- `tenant_id` - (Optional) The owner of the policy. Required if admin wants to create a policy for another project. Changing this creates a new policy.
- `description` - (Optional) The human-readable description for the policy. Changing this updates the description of the existing policy.
- `auth_algorithm` - (Optional) The authentication hash algorithm. Valid values are sha1, sha256, sha384, sha512. Default is sha1. Changing this updates the algorithm of the existing policy.
- `encapsulation_mode` - (Optional) The encapsulation mode. Valid values are tunnel and transport. Default is tunnel. Changing this updates the existing policy.
- `encryption_algorithm` - (Optional) The encryption algorithm. Valid values are 3des, aes-128, aes-192 and so on. The default value is aes-128. Changing this updates the existing policy.
- `pfs` - (Optional) The perfect forward secrecy mode. Valid values are Group2, Group5 and Group14. Default is Group5. Changing this updates the existing policy.
- `transform_protocol` - (Optional) The transform protocol. Valid values are ESP, AH and AH-ESP. Changing this updates the existing policy. Default is ESP.
- `lifetime` - (Optional) The lifetime of the security association. Consists of Unit and Value.
  - `unit` - (Optional) The units for the lifetime of the security association. Can be either seconds or kilobytes. Default is seconds.
  - `value` - (Optional) The value for the lifetime of the security association. Must be a positive integer. Default is 3600.
- `value_specs` - (Optional) Map of additional options.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `description` - See Argument Reference above.
- `auth_algorithm` - See Argument Reference above.
- `encapsulation_mode` - See Argument Reference above.
- `encryption_algorithm` - See Argument Reference above.
- `pfs` - See Argument Reference above.
- `transform_protocol` - See Argument Reference above.
- `lifetime` - See Argument Reference above.
  - `unit` - See Argument Reference above.
  - `value` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

Import

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

```
$ terraform import openstack_vpnaas_ipsec_policy_v2.policy_1 832cb7f3-59fe-40cf-8f64-8350ffc03272
```

openstack_vpnaas_service_v2

Manages a V2 Neutron VPN service resource within OpenStack.

Example Usage

```
resource "openstack_vpnaas_service_v2" "service_1" {
  name           = "my_service"
  router_id      = "14a75700-fc03-4602-9294-26ee44f366b3"
  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 VPN service. If omitted, the `region` argument of the provider is used. Changing this creates a new service.
- `name` - (Optional) The name of the service. Changing this updates the name of the existing service.
- `tenant_id` - (Optional) The owner of the service. Required if admin wants to create a service for another project. Changing this creates a new service.
- `description` - (Optional) The human-readable description for the service. Changing this updates the description of the existing service.
- `admin_state_up` - (Optional) The administrative state of the resource. Can either be up(true) or down(false). Changing this updates the administrative state of the existing service.
- `subnet_id` - (Optional) SubnetID is the ID of the subnet. Default is null.
- `router_id` - (Required) The ID of the router. Changing this creates a new service.
- `value_specs` - (Optional) Map of additional options.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `router_id` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.

- `subnet_id` - See Argument Reference above.
- `status` - Indicates whether IPsec VPN service is currently operational. Values are ACTIVE, DOWN, BUILD, ERROR, PENDING_CREATE, PENDING_UPDATE, or PENDING_DELETE.
- `external_v6_ip` - The read-only external (public) IPv6 address that is used for the VPN service.
- `external_v4_ip` - The read-only external (public) IPv4 address that is used for the VPN service.
- `description` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

Import

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

```
$ terraform import openstack_vpnaas_service_v2.service_1 832cb7f3-59fe-40cf-8f64-8350ffc03272
```

openstack_vpnaas_site_connection_v2

Manages a V2 Neutron IPSec site connection resource within OpenStack.

Example Usage

```
resource "openstack_vpnaas_site_connection_v2" "conn_1" {
  name              = "connection_1"
  ikepolicy_id      = "${openstack_vpnaas_ike_policy_v2.policy_2.id}"
  ipsecpolicy_id    = "${openstack_vpnaas_ipsec_policy_v2.policy_1.id}"
  vpnservice_id     = "${openstack_vpnaas_service_v2.service_1.id}"
  psk               = "secret"
  peer_address      = "192.168.10.1"
  local_ep_group_id = "${openstack_vpnaas_endpoint_group_v2.group_2.id}"
  peer_ep_group_id  = "${openstack_vpnaas_endpoint_group_v2.group_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 IPSec site connection. If omitted, the `region` argument of the provider is used. Changing this creates a new site connection.
- `name` - (Optional) The name of the connection. Changing this updates the name of the existing connection.
- `tenant_id` - (Optional) The owner of the connection. Required if admin wants to create a connection for another project. Changing this creates a new connection.
- `description` - (Optional) The human-readable description for the connection. Changing this updates the description of the existing connection.
- `admin_state_up` - (Optional) The administrative state of the resource. Can either be up(true) or down(false). Changing this updates the administrative state of the existing connection.
- `ikepolicy_id` - (Required) The ID of the IKE policy. Changing this creates a new connection.
- `vpnservice_id` - (Required) The ID of the VPN service. Changing this creates a new connection.
- `local_ep_group_id` - (Optional) The ID for the endpoint group that contains private subnets for the local side of the connection. You must specify this parameter with the peer_ep_group_id parameter unless in backward- compatible mode where peer_cidrs is provided with a subnet_id for the VPN service. Changing this updates the existing connection.
- `ipsecpolicy_id` - (Required) The ID of the IPsec policy. Changing this creates a new connection.
- `peer_id` - (Required) The peer router identity for authentication. A valid value is an IPv4 address, IPv6 address, e-mail address, key ID, or FQDN. Typically, this value matches the peer_address value. Changing this updates the existing policy.

- `peer_ep_group_id` - (Optional) The ID for the endpoint group that contains private CIDRs in the form < net_address > / < prefix > for the peer side of the connection. You must specify this parameter with the local_ep_group_id parameter unless in backward-compatible mode where peer_cidrs is provided with a subnet_id for the VPN service.
- `local_id` - (Optional) An ID to be used instead of the external IP address for a virtual router used in traffic between instances on different networks in east-west traffic. Most often, local ID would be domain name, email address, etc. If this is not configured then the external IP address will be used as the ID.
- `peer_address` - (Required) The peer gateway public IPv4 or IPv6 address or FQDN.
- `psk` - (Required) The pre-shared key. A valid value is any string.
- `initiator` - (Optional) A valid value is response-only or bi-directional. Default is bi-directional.
- `peer_cidrs` - (Optional) Unique list of valid peer private CIDRs in the form < net_address > / < prefix > .
- `dpd` - (Optional) A dictionary with dead peer detection (DPD) protocol controls.
    - `action` - (Optional) The dead peer detection (DPD) action. A valid value is clear, hold, restart, disabled, or restart-by-peer. Default value is hold.
    - `timeout` - (Optional) The dead peer detection (DPD) timeout in seconds. A valid value is a positive integer that is greater than the DPD interval value. Default is 120.
    - `interval` - (Optional) The dead peer detection (DPD) interval, in seconds. A valid value is a positive integer. Default is 30.
- `mtu` - (Optional) The maximum transmission unit (MTU) value to address fragmentation. Minimum value is 68 for IPv4, and 1280 for IPv6.
- `value_specs` - (Optional) Map of additional options.

Attributes Reference

The following attributes are exported:

- `region` - See Argument Reference above.
- `name` - See Argument Reference above.
- `tenant_id` - See Argument Reference above.
- `admin_state_up` - See Argument Reference above.
- `description` - See Argument Reference above.
- `dpd` - See Argument Reference above.
- `psk` - See Argument Reference above.
- `initiator` - See Argument Reference above.
- `peer_address` - See Argument Reference above.
- `peer_id` - See Argument Reference above.
- `peer_cidrs` - See Argument Reference above.

- `mtu` - See Argument Reference above.
- `local_id` - See Argument Reference above.
- `peer_ep_group_id` - See Argument Reference above.
- `ipsecpolicy_id` - See Argument Reference above.
- `vpnservice_id` - See Argument Reference above.
- `ikepolicy_id` - See Argument Reference above.
- `value_specs` - See Argument Reference above.

Import

Site Connections can be imported using the `id`, e.g.

```
$ terraform import openstack_vpnaas_site_connection_v2.conn_1 832cb7f3-59fe-40cf-8f64-8350ffc03272
```