Vultr Guide

Ansible offers a set of modules to interact with Vultr cloud platform.

This set of module forms a framework that allows one to easily manage and orchestrate one's infrastructure on Vultr cloud platform.

Requirements

There is actually no technical requirement; simply an already created Vultr account.

Configuration

Vultr modules offer a rather flexible way with regard to configuration.

Configuration is read in that order:

- Environment Variables (eg. `VULTR_API_KEY`, `VULTR_API_TIMEOUT`)
- File specified by environment variable `VULTR_API_CONFIG`
- `vultr.ini` file located in current working directory
- `$HOME/.vultr.ini`

Ini file are structured this way:

```
[default]
key = MY_API_KEY
timeout = 60

[personal_account]
key = MY_PERSONAL_ACCOUNT_API_KEY
timeout = 30
```

If `VULTR_API_ACCOUNT` environment variable or `api_account` module parameter is not specified, modules will look for the section named "default".

Authentication

Before using the Ansible modules to interact with Vultr, ones need an API key. If one doesn't own one yet, log in to Vultr go to Account, then API, enable API then the API key should show up.

Ensure you allow the usage of the API key from the proper IP addresses.

Refer to the Configuration section to find out where to put this information.

To check that everything is working properly run the following command:

```
#> VULTR_API_KEY=XXX ansible -m vultr_account_info localhost
localhost | SUCCESS => {
  "changed": false,
  "vultr_account_info": {
      "balance": -8.9,
      "last_payment_amount": -10.0,
      "last_payment_date": "2018-07-21 11:34:46",
      "pending_charges": 6.0
  },
  "vultr_api": {
      "api_account": "default",
      "api_endpoint": "https://api.vultr.com",
      "api_retries": 5,
      "api_timeout": 60
  }
}
```

If a similar output displays then everything is setup properly, else please ensure the proper `VULTR_API_KEY` has been specified and that Access Control on Vultr > Account > API page are accurate.

Usage

Since Vultr offers a public API, the execution of the module to manage the infrastructure on their platform will happen on localhost. This translates to:

```
---
- hosts: localhost
  tasks:
```

```
    - name: Create a 10G volume
      vultr_block_storage:
        name: my_disk
        size: 10
        region: New Jersey
```

From that point on, only your creativity is the limit. Make sure to read the documentation of the available modules.

Dynamic Inventory

Ansible provides a dynamic inventory plugin for Vultr. The configuration process is exactly the same as the one for the modules.

To be able to use it you need to enable it first by specifying the following in the `ansible.cfg` file:

```
[inventory]
enable_plugins=vultr
```

And provide a configuration file to be used with the plugin, the minimal configuration file looks like this:

```
---
plugin: vultr
```

To list the available hosts one can simply run:

```
#> ansible-inventory -i vultr.yml --list
```

For example, this allows you to take action on nodes grouped by location or OS name:

```
---
- hosts: Amsterdam
  tasks:
    - name: Rebooting the machine
      shell: reboot
      become: True
```

Integration tests

Ansible includes integration tests for all Vultr modules.

These tests are meant to run against the public Vultr API and that is why they require a valid key to access the API.

Prepare the test setup:

```
$ cd ansible # location the ansible source is
$ source ./hacking/env-setup
```

Set the Vultr API key:

```
$ cd test/integration
$ cp cloud-config-vultr.ini.template cloud-config-vultr.ini
$ vi cloud-config-vultr.ini
```

Run all Vultr tests:

```
$ ansible-test integration cloud/vultr/ -v --diff --allow-unsupported
```

To run a specific test, for example vultr_account_info:

```
$ ansible-test integration cloud/vultr/vultr_account_info -v --diff --allow-unsupported
```