terrible Documentation

Release 0.1.2

Michael J. Palmer

Aug 21, 2017

Contents

Contents:

CHAPTER 1

terrible

Terrible (TERRaform to ansIBLE) creates dynamic Ansible inventory from Terraform state.

- Free software: Apache Software License 2.0
- Documentation: https://terrible.readthedocs.io.

Installation

To install terrible, run this command in your terminal.:

```
$ pip install terrible
```

Once installed a symbolic link or shell script can be added to the Ansible inventory directory.

Symbolic link.:

```
$ ln -s /path/to/terrible inventory/terrible
```

Simple shell script wrapper.:

```
#!/usr/bin/env bash
terrible "$@"
```

Usage

```
Usage: terrible [OPTIONS] <root_dir>

  Terrible extracts Ansible inventory data from Terraform state. The
  <root_dir> is relative to the directory where Ansible is executed but
  defaults to ./terraform in the current directory.
```

```
Options:
  --host TEXT  Show varibles for single host
  --list       List all variables
  --nometa     Remove _meta from output
  --pretty     Make json look pretty
  --help       Show this message and exit.
```

Features

Terraform Resources:

- VMware vSphere (vsphere_virtual_machine)
- AWS Instance (aws_instance)

Common Parameters

These can be specified by all resources. Uniq configuration details are documented in specific sections below.

ansible_user (Optional) The user that Ansible will connect to the host. Defaults to root if not specified.

ansible_group (Optional) The inventory group associated with the resource. (Add default All group?)

ansible_host (Optional) The host that Ansible will connect to. VMware defaults to IP of 1st interface, `network_interface:0` but if can be overwriten to an specific IP. AWS defaults to `public_ip` and configuralbe to `private_ip`. (TODO: Add test and error condition for values)

VMware

When defining Terraform `vsphere_virtual_machine` resource use the `custom_configuration_parameters` block to set Ansible parameters.

Configuration example:

```
custom_configuration_parameters {
  ansible_group = "api-gateway"
  ansible_user = "ansible"
  ansible_host = "192.168.52.101"
}
```

AWS

When defining a Terraform `aws-instance` resource use tags to set Ansible parameters.

ansible_ssh_private_key_file The key used to connect to AWS instance. The value is the path to the private key that matches the defined AWS instance key_name. Defaults to the value of `key_name + .pem`. EXAMPLE: If your AWS instance key_name is `terraform` then Ansible would look in the current working directory for terraform.pem

Configuration example:

```
tags {
  Name = "app1-aws"
  ansible_groups = "webapp"
  ansible_user = "ansible"
  ansible_host = "private_ip"
  ansible_ssh_private_key_file = "aws-keys/webapp-terraform.pem"
}
```

Directory Layout

By default, Terrible looks for the `terraform` inside the Ansible playbook root directory.:

```
.
- ansible.cfg
- inventory
|    - group_vars
|    - terrible
- playbooks
|    - site.yml
- requirements.yml
- roles
|    - example_role
- terraform
     - terraform.tf
     - terraform.tfstate
     - terraform.tfvars
     - variables.tf
```

Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

This package was greatly influenced by the sean-abbott/terraform.py project.

CHAPTER 2

Installation

Stable release

To install terrible, run this command in your terminal:

```
$ pip install terrible
```

This is the preferred method to install terrible, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

From sources

The sources for terrible can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/palmertime/terrible
```

Or download the tarball:

```
$ curl  -OL https://github.com/palmertime/terrible/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use terrible in a project:

```
import terrible
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/palmertime/terrible/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

terrible could always use more documentation, whether as part of the official terrible docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/palmertime/terrible/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *terrible* for local development.

1. Fork the *terrible* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/terrible.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv terrible
$ cd terrible/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 terrible tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/palmertime/terrible/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_terrible
```

CHAPTER 5

Credits

Development Lead

- Michael J. Palmer <palmertime@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.1.2 (2017-08-21)

- bump version testing

0.1.1 (2017-08-21)

- Support for `aws_instance` and `vsphere_virtual_machine` Terraform resources.
- Ansible parameters available: ansible_user, ansible_host, ansible_ssh_private_key_file
- Define Terraform resource to ansible group.

0.1.0 (2017-08-10)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search