cinch Documentation

Release

RedHatQE

January 19, 2017

Contents

CHAPTER 1

Users

CHAPTER 2

Getting Started

At its core, this softare is nothing more than a collection of Ansible playbook scripts for configuring up a system. Any knowledge that you have that is applicable to the broad spectrum of Ansible usage is applicable here. You can opt to install Ansible from your favorite package manager, or you can use the version that is pinned in the requirements.txt file and has been tested with the software.

Before concluding there is a bug in these playbooks, make sure that the version of Ansible you are using is the same as the version in the requirements.txt file and that you have ensured there are no alterations from that version. It is not intended or guaranteed that any changes from the stock version of Ansible that has been tested should work.

CHAPTER 3

Requirements

To setup your environment, you need to install Ansible. Since Ansible is primarily distributed as a Python package, it is suggested that you use pip to install Ansible on your system. You are welcome to try and use the version that is installed by your favorite package manager, but be sure that you are using a version at least as new as the version pinned in requirements.txt.

It is recommended that you install Ansible from Pip using a virtualenv, as is the best practices recommendations for most Python packages that are available from PyPI. In order to build and install Ansible, you will need to install the following system packages

- gcc or appropriate system compiler
- OpenSSL development package
- libyaml development package
- virtualenv package

Use your system package manager to install these packages, if they are not already present. Note that you will need to install the development version of the libraries, as Pip will attempt to build wrappers around some of those libraries during its install of Ansible and dependencies.

CHAPTER 4

Installation

Once the system level packages are installed, you can use one of two different methods to install Ansible and any other Python dependencies

4.1 Packaged Script

To use the existing packaged script, simply invoke bin/ensure_virtualenv.sh, and it will create the necessary virtualenv and install the Python packages using the pip command in that virtualenv.

Activate the virtualenv by calling `source .venv/bin/activate` for a bash environment, or the otherwise appropriate activate script for your preferred shell environment. Typing `deactivate` will shut down the virtualenv.

4.2 Manual Install

For a manual install, follow these steps

- Create a virtualenv of your choice
- Activate the newly created virtualenv
- Execute a `pip install -r requirements.txt` on that virtualenv

At this point, that virtualenv should be setup for using Ansible to run these playbooks.

CHAPTER 5

Execution

Execution of this softare requires configuring an Ansible inventory that points at the jenkins_master and jenkins_slave hosts that you want configured. Use normal methods for setting group_vars and host_vars within the inventory or its associated folders that suits your own needs and preferences.

While most default settings should be functional, there are lots of options configured in the various default/main.yml files within the various roles folders. Check in those files for more details on specific options that can be set and a description of what they each mean.

See a few examples of such in either the inventory/ folder or inside of the various vagrant/ subfolders where known good working environments are configured for development use.

The path inventory/local is excluded from use by the project and can be leveraged for executing and storing your own local inventories, if the desire arises. There is even a shell script in bin/run_jenkins_local.sh that will execute ansible-playbook from the .venv/ virtualenv and point it to the inventory/local/hosts file to make executing against your own environment as easy as a single command.

CHAPTER 6

Support

The playbooks should support, minimally, CentOS and RHEL versions 7+. If you encounter difficulties in those environments, please file bugs. There should be no configuration necessary for a CentOS host, and a RHEL host requires only that you configure the base URL for your local RHEL repository collection. See documentation in the appropriate roles for details on that configuration.

CHAPTER 7

Development

CHAPTER 8

ENVIRONMENTS

Development occurs targeting each of the specific host environments that are supported. The default development environment and targeted host is the latest version of CentOS.

The fastest way to get yourself up and running is to leverage the Vagrant machines held within the top-level vagrant folder. These are named according to the roles that each one is designed to exercise.

8.1 Install

To run the software locally, you need a few basic pieces of software installed. The following packages for Fedora need to be installed, minimally, or the equivalent packages for your distribution:

- python-virtualenv
- gcc
- redhat-rpm-config
- openssl-devel
- libvirt-devel
- libyaml-devel
- vagrant

The only software actually required to run the playbooks is Ansible and its dependencies. To install the versions that have been tested as working with this system, use the requirements.txt file in the top of the project to install all the necessary software into a Python virtualenv. Alternatively, invoke the script bin/ensure_virtualenv.sh to install the Python dependencies into the folder .venv/

If installing manually, you can activate your Python virtualenv of choice and issue the command `pip install -r requirements.txt`

8.2 Execution

Once all of these depenencies are fulfilled, there are a number of folders under the top level vagrant/ directory that contain minimally a Vagrantfile and a hosts file. The Vagrantfile can be used to issue the command "vagrant up" from within that directory to spin up a collection of machines, and the hosts file can be used as the inventory input for running the Ansible playbooks against the spun up Vagrant VMs.

Inside of each of those folders should also be a pair of shell scripts. These scripts are named "full_cycle.sh" and the other "configure.sh". full_cycle.sh will execute a `vagrant destroy -f && vagrant up` followed by

invoking the same commands that configure.sh invokes. configure.sh will ensure a virtualenv has been created and the requirements.txt file has been installed to that environment. It will then execute ansible-playbook from that virtualenv against the hosts inventory file in the folder.

8.3 Customization

Both configure.sh and full_cycle.sh will pass any arguments they receive, unaltered, through to the ansible-playbook invocation at the end of the command line. Thus, if any extra arguments need to be appended to the ansible-playbook invocation, for instance passing in command-line variable overrides or setting verbose flags, they can be added to the end of the invocation for the shell script.

CHAPTER 9

Indices and tables

- genindex
- search