cinch Documentation

RedHatQE

Oct 01, 2018

Contents

CHAPTER 1

Users

1.1 Quick Start

If you would rather not be bothered with the boring, gritty details, the following quick steps will get you up and going with a Cinch installation, after which you can jump down to the section on how to run Cinch, and skip the installation paragraphs that immediately follow this one.

1.1.1 Fedora 25+

Execute the following commands

```
sudo dnf install -y libvirt-devel python-virtualenv libyaml-devel
openssl-devel libffi-devel gcc redhat-rpm-config
```

```
virtualenv cinch && source cinch/bin/activate
```

```
pip install cinch
```

1.1.2 RHEL/CentOS

See the sample Ansible playbook for steps to install Cinch into a virtualenv in RHEL/CentOS, as there are some additional steps required in these systems. Running that playbook as-is on your local system will result in a virtualenv living at `/var/lib/jenkins/opt/cinch` that will contain the latest version of Cinch.

1.1.3 Ubuntu

```
apt-get install -y libvirt-dev python-virtualenv libyaml-dev openssl
libffi-dev gcc python-dev libssl-dev
```

```
virtualenv cinch && source cinch/bin/activate
```

On older versions of Ubuntu (like 14.04) you should update pip (usually these older systems come with a version of pip such as 1.5.4 instead of version 9+)

```
pip install -U pip
```

On all systems, continue on with the installation of Cinch itself

```
pip install cinch
```

Note: After Cinch is installed with the above quick start methods, you can jump down to the section about running Cinch to get a look at basic documentation on how the software should be used.

1.2 Getting Started

At its core, this software is nothing more than a collection of Ansible playbook scripts for configuring 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 specified in **setup.py**

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 **setup.py** 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.

1.3 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 **setup.py**.

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:

Note: If you install cinch via **pip**, a supported version of Ansible will be brought in as a dependency.

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

Use your system package manager to install these packages, if they are not already present.

Note: 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.

Here is an example of installing required system level packages for Fedora 25:

1.4 Installation

Once the system level packages are installed, you can install cinch using **pip** (virtualenv strongly recommended):

1.4.1 Fedora

```
virtualenv cinch && source cinch/bin/activate
pip install cinch
```

1.4.2 RHEL7 and CentOS7

RHEL7 and derivatives offer older versions of Python packaging tools that are incompatible with some cinch dependencies. To work-around this issue, we have provided an Ansible playbook that will install a newer version of the necessary Python packaging tools to allow for installation on RHEL7. This playbook is intended for use on Jenkins masters and will install cinch and linchpin into a virtualenv at **/var/lib/jenkins/opt/**. For convenience, an optional Jenkins Job Builder template is provided and will create a Jenkins job that will run the aforementioned playbook on your Jenkins master.

1.5 Execution

1.5.1 With linchpin

If you'd like to automate the process of provisioning a host to later configure with cinch, the linchpin project can be used for this task. linchpin can dynamically generate an Ansible inventory file that cinch can consume for host configuration. In the following steps we will outline how to configure cinch-specific values within a linchpin workspace.

Note: For linchpin topology and workspace examples, including various host environments, see the linchpin documentation.

Create a layout file by saving the following example template as **/path/to/linchpin/workspace/layouts/mylayout.yml** and edit to taste based on your cinch role requirements. In this example we configure a RHEL7 Jenkins slave:

```
---
inventory_layout:
  hosts:
    cinch-group:
      count: 1
      host_groups:
        - rhel7
        - certificate_authority
        - repositories
        - jenkins_slave
```

Create an Ansible **group_vars** file by saving the following example template as **/path/to/linchpin/workspace/inventories/group_vars/all** and edit to taste based on your desired configuration parameters. In this example we configure a RHEL7 Jenkins slave to attach to a Jenkins master which requires authentication, along with some installed certificate authorities and repositories:

```
---
ansible_user: root
ansible_private_key_file: "{{ inventory_dir }}/../keystore/ssh-key"
ansible_connection: ssh
# Add URLs from which to download CA certificates for installation
certificate_authority_urls:
  - https://example.com/ca1.crt
  - https://example.com/ca2.crt
# Base URL for repository mirror
rhel_base: http://example.com/content/dist/rhel/server/7/7Server
jenkins_master_url: 'http://jenkins.example.com' # URL to Jenkins master for the␣
↪slave to connect to
jslave_name: 'cinch-slave'
jslave_label: 'cinch-slave'
# If your Jenkins master requires authentication to connect a slave,
# add credentials via the two variables below.  If anonymous users can
# connect slaves to the master, do not include the following two
# variables in this layout file.
jenkins_slave_username: 'automation-user'
jenkins_slave_password: 'jenkinsAPItoken'
```

Finally, If you'd like to automate this process in Jenkins, please see our example Jenkins Job Builder workflow template for guidance on putting it all together.

1.5.2 Manual

Execution of this software 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.

See also:

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. *These files are heavily commented, and serve as the best source of documentation for the way that cinch configures a system. If you feel the need to modify cinch playbooks directly, first check to see if the behavior you want is configurable via the provided Ansible variables.*

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.

The cinch project can be used as a standard Ansible project, by running **ansible-playbook** and calling **site.yml** for Jenkins master or slave configuration and **teardown.yml** for removing a Jenkins slave from a Jenkins master.

For convenience, we also provide CLI wrappers for these tasks. These wrappers simplify the task of finding and running the desired cinch playbooks for configuration or teardown, and also can optionally pass through any additional CLI arguments to the 'ansible-playbook' command that you may need.

The following commands are available:

- `cinch` - runs the **site.yml** playbook to configure a Jenkins master or slave
- `teardown` - runs the **teardown.yml** playbook to disconnect a Jenkins slave

Use the `-h` or `--help` arguments for the CLI wrappers to get further info.

1.6 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 2

Development

2.1 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.

2.2 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. The other packages listed above are required only to install and build Ansible and its dependencies, such as PyYAML. Thus, if you are looking to package Cinch for a new distribution, the above packages, less vagrant, are a good starting place for build dependencies.

If installing manually, you can activate your Python virtualenv of choice and issue the command `pip install /path/to/cinch`. As a developer, if you plan to make changes to Cinch, then use pip in the local editable mode by issuing the command `pip install -e /path/to/cinch` instead.

2.3 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. The Vagrantfile can be used to issue the command "vagrant up" from within that directory to spin up a collection of machines, against which the cinch playbooks will be automatically executed. Consult the README in each directory for more information about which machines will be created out of that directory, and for any information that the user might need to supply.

Some of the Vagrantfile values will need to be supplied by the user, specifically any values related to RHEL repository URLs as there is no public version of those repositories available. Other values should all be provided from within those directories already.

Merely issuing the command `vagrant up` should bring up the VMs for each environment you configure. For the most part, it should be possible to run each environment on your local system, but there is the potential that having multiple environments running at the same time on the same host could result in collissions between the IP addresses of the hosts. It certainly would lead to provided URLs in the README files being incorrect.

CHAPTER 3

Docker Image

For users who do not want to provision an entire system to run a Jenkins slave there exists a Docker image which can quickly get a Jenkins Swarm connected instance to run.

3.1 Source Image

For every release of cinch that is made, a version of the Docker container is pushed to Docker Hub. Multiple tags are pushed for each Cinch release. They are named by combining source information image along with the version of Cinch used to build them.

Currently there are images built off of

- centos:7
- centos:6

These get tagged into the Cinch image repository as

- redhatqecinch/jenkins_slave:cent7-0.5.2
- redhatqecinch/jenkins_slave:cent6-0.5.2

This indicates two images, one based on the centos:7 image and one based off the centos:6 image. Both of them are built by the version 0.5.2 release of Cinch.

3.2 Image Options

As with the rest of Cinch, there are some customizable image options that a user must supply before the image will work with your infrastrucutre. However, unlike using the Ansible-based solution to create your own system, there are far fewer options. Other than the following options, all builds of the Cinch Docker images utilize all default values for a Cinch slave instance.

There are two variables that the user is required to provide before the image will run properly. Those are

Environment Variable	Explanation
JENKINS_MASTER_URL	The URL to the Jenkins master instance that this slave should connect to
JSLAVE_NAME	The name this slave will be given on the Master node
JSLAVE_LABEL	The Jenkins label this slave will receive, which will be matched against jobs requiring certain labels for execution
JSWARM_EXTRA_ARGS	Additional command-line arguments to pass to the JSwarm client in the iamge

If the container image is run directly from the Docker command line, these options may be passed through *docker*'s -e option. When running the image in Kubernetes or OpenShift, use that system's methods for passing in environment variables to the iamge.

3.3 Customizing the Image

Instead of running the base image provided, a group could choose to use a Dockerfile to extend the base image provided to do such things as install custom software, edit configurations, etc. If that is the case, then the environment variables can absolutely be preset within the Dockerfile using its ENV command, as with any other environment variable.

Extending the image in this way could simplify deployment, as the image could include information such as the Jenkins Master URL already configured to connect to the organization's Jenkins instance. Likewise, different slave images could be pre-populated with packages and slave labels for building different types of software or running different types of tasks. As nothing more than a standard Docker image, the provided images can be made fully extensible.

One note is that the image is set to run all commands as the user "jenkins". If the image is being extended, then it might be necessary to set the USER command in the extending Dockerfile to "USER root" if system software is being installed.

CHAPTER 4

User Files

4.1 Motivation

Anyone using Cinch to provision either a Jenkins master or slave may have the need to perform configuration to the system that exceeds the ability of Cinch to reasonably include support for within these playbooks. These could cover nearly any aspect of system administration, monitoring, configuration, and setup. For such a case, it is recommended that the user leverage the ability of Ansible to file a host into multiple different inventory groups, and private configuration be stored in private playbooks. Then those playbooks can be executed either before or after (or both) the Cinch playbooks are executed.

However, there are a few basic system administration tasks that are general enough, and simple enough, that Cinch has opted to support those features to assist in the configuration of a Jenkins master. In addition to supporting the ability to setup Yum/DNF repositories during configuration and configure certificate authority chains, both of which are important to installing the packages required by Cinch and to configure SSL options for Jenkins, another feature supported by Cinch is the ability to upload arbitrary files from the local system where Ansible is being hosted to the remote system being configured.

4.2 Mechanisms

Each Ansible host, or group, can have defined values of files to upload to the remote hosts. These uploads happen at two different points during the execution of Cinch. The first set of uploads occurs before any Cinch plays have been executed except for verifying the host is reachable. This means that none of the Cinch-related configurations will be available during this upload run, unless they have previously been configured. This includes things like the "jenkins" system user, configured repositories, certificate authorities, etc. The second run happens at the very end - after both the master and any slaves have been configured and are up and running. However, at this point, all such configurations, users, etc are already present on the system.

Thus, it is important to realize a file cannot be uploaded to be owned by the Jenkins user before the Jenkins user is created. If it is necessary to upload a file as that user before the Jenkins service starts on a configured host, then it will be necessary to use external playbooks or other methods to ensure proper behavior.

4.3 Configuration

Configuring uploads either before or after a Cinch run is straightforward. Simply override the values of the arrays "pre_upload_files" and "post_upload_files" in the Ansible host or group configurations for all hosts that require such a feature.

These arrays require identical structures. Each element in the array should be an object hash with certain values defined. Those values are listed below:

value	required?
src	yes
dest	yes
owner	no
group	no
mode	no

Example:

```
pre_upload_files:
  - src: /home/deployuser/somehost/ssl.key
    dest: /etc/apache2/ssl/ssl.key
    mode: 0600
post_upload_files:
  - src: /home/deployuser/somehost/ssh
    dest: /var/lib/jenkins/.ssh
    owner: jenkins
    mode: 0600
```

Each of these values is passed directly into the Ansible module called copy. Refer to that module's documentation for information about the structure and values that are permitted to be passed into these values. Note, especially, that this module can be used to upload whole directories in addition to individual files.

If the need arises to support more of the options of that module, adding that support to Cinch can be done. Please just open an issue in the GitHub Issue Tracker detailing the requested functionality.

CHAPTER 5

Common Config Options

5.1 All Options

To get the latest up-to-date list of all availble options in Cinch, consult the files for each Ansible role in the cinch/roles/<role_name>/defaults/main.yml files in the code base. Every variable should be documented, along with the default value given.

5.2 Jenkins Plugins

Cinch configures what has been deemed and tested as a reasonable baseline set of Jenkins plugins. Typically it will not be necessary to alter or remove elements from this list. The current list can be found in the file cinch/files/jenkins-plugin-lists/default.txt. Opening this file will give a list of plugins, one per line. A specific version of a plugin can be specified by a line that reads "myplugin==1.2.3" and will install specifically version 1.2.3 of that plugin.

If the set of default plugins is not acceptable to a user, they can override the list by defining the variable jenkins_plugins in their host or group vars for a Cinch run to include the items they want. This variable is an array of strings, each string being the equivalent of one line from the default.txt file.

If a user only wants to add some plugins that are not present in the default set, without completely overriding the set, this can be accomplished by adding entries to jenkins_extra_plugins in the same format as entries in the jenkins_plugins variable. This allows the user to install more plugins than the default, without needing to worry about falling out of sync with the default set of plugins

CHAPTER 6

Maintainers

cinch contains automation to aid in the process of creating releases on GitHub and PyPI, driven by Travis CI.

6.1 Release Procedure

As a maintainer, to create a release of cinch, follow this checklist:

- Ensure version has been bumped in **setup.py**, following Semantic Versioning guidelines
- Ensure significant changes are listed in the **CHANGELOG**
- Merge desired changes, including the above checklist items to the **master** branch
- The release is based on git tags, and the following steps can be followed to tag the release, given that the upstream remote in your git config is named **upstream** (adjust git remote and version number as necessary):

```
git fetch upstream master
git merge upstream/master
git tag v0.9.0
git push --tags upstream master
```

- It will take about fifteen minutes or so for the automation to add the release to PyPI, and while you wait, manually copy the release notes from the **CHANGELOG** to the GitHub releases page. This step is something that could possibly be automated in the future.

CHAPTER 7

Indices and tables

- genindex
- search