OpenStack-Ansible Documentation: python_venv_build role

Release 0.1.0.dev204

OpenStack-Ansible Contributors

Mar 10, 2023

CONTENTS

openstack community project cii best practices passing

CHAPTER

ONE

OPENSTACK-ANSIBLE PYTHON_VENV_BUILD

This Ansible role prepares a python venv for use within the OpenStack-Ansible project, but it may be used for other projects as well.

The role requires the following to be present prior to execution:

- virtualenv >= 1.10 (to support using the never-download option)
- pip >= 7.1 (to support using the constraints option) in the virtualenv once it has been created.

1.1 Use-cases

This role is built for the following use-cases:

1. Using a build host (a.k.a. repo server):
   - Build python wheels on a repo server with a given list of python packages.
   - Prepare a requirements.txt and constraints.txt file on the repo server, and use them to ensure that the build and installation processes are both consistent and idempotent.
   - On the build host, install the distribution packages required at build time.
   - On any number of target hosts, create a virtualenv and install these built wheels into it using the pip `--find-links` option.
   - On any number of target hosts, install the distribution packages required at run time.
   - Re-use previously built wheels to speed up any subsequent builds..
2. Not using a build host:
   - On any number of target hosts, create a virtualenv, then locally install the distribution packages required at build and run time, then locally compile and install the given list of python packages.
   - This negates the need for a repo server, but takes longer due to the increased number of dependencies to install and the compilation happening on every target host.
   - The only situation where a build host provides no benefit is where there is only a single target host (with no containers) and none of the packages installed into the venv will be used again for any other venvs built by this role on the same host.

It may be useful to review the Python Build/Install Process Simplification specification to understand the background that led to the creation of this role.

1.2 Process

1. Pre-requisites are checked.
2. If wheel building is enabled, and there is a repo server in the environment, then the following happens on the repo server:
   a. The distribution packages required to execute the python wheel compile are installed.
   b. A set of requirements and source-constraints for the venv are compiled for pip to use when building the wheels. These are also used to determine whether there are changes to either for the purpose of idempotence.
   c. The python wheels are compiled, and an install-time constraints file is created. The install-time constraints file has the list of python packages with their versions - this differs from the source-constraints which may contain git SHAs.
3. The installation of the python packages then commences on the target hosts:
   a. If the wheel build was enabled:
      i. Only the distribution packages required at runtime by the python packages are installed.
      ii. A python venv is created at `venv_install_destination_path`.
      iii. The requirements and constraints files for the venv are prepared in the venv path.
      iv. The python packages are installed from the wheels on the repo server using pips `--find-links` option to ensure that they are preferred above the default pypi index.
      v. If there are any `venv_packages_to_symlink` then the appropriate python libraries installed into the system from those packages will be symlinked into the virtualenv. This provides for python libraries which have a tight coupling with C bindings which may not be portable as a wheel.
   b. If the wheel build was *not* enabled:
      i. The distribution packages required for compiling and at runtime by the python packages are installed.
      ii. A python venv is created at `venv_install_destination_path`.
      iii. The requirements and constraints files for the venv are prepared in the venv path. The constraints file in this case would contain the same content as the source-constraints file on the repo server where there is one.
      iv. The python packages are installed from the default pip index. During the installation pip will do a git clone and build from it for any packages that have a git SHA as a constraint.
      v. If there are any `venv_packages_to_symlink` then the appropriate python libraries installed into the system from those packages will be symlinked into the virtualenv. This provides for python libraries which have a tight coupling with C bindings which may not be portable as a wheel.
4. If any `venv_facts_when_changed` are set, then they are implemented on the target host in `/etc/ansible/facts.d`.

1.3 Default variables

```
#
# Required variables
#

# The path where venvs are extracted to
# on the target host during an install, for example:
# venv_install_destination_path: "/openstack/venvs/myvenv"

#
# Optional variables
#

# Distribution packages which must be installed
# on all hosts when building python wheels.
venv_build_base_distro_package_list: "{{ _venv_build_base_distro_package_
↪list[ansible_facts['os_family'] | lower] }}"

# Distribution packages which must be installed
# on the host for the purpose of building the
# python wheels.
venv_build_distro_package_list: []

# Distribution packages which must be installed
# on the host when installing the venv.
venv_install_distro_package_list: []

# Set the package install state for packages
# Options are 'present' and 'latest'
venv_distro_package_state: "latest"
venv_pip_package_state: "latest"

# The time in seconds that the distribution package
# cache is valid for. This is only used by the apt
# package manager.
venv_distro_cache_valid_time: 600

# Default python packages which will be installed
# into every venv.
venv_default_pip_packages: []

# Python packages which must be installed
# into the venv.
venv_pip_packages: []

# Don't use the site-wide PIP configuration file when
# upgrading PIP (some operating systems have issued
# with upgrades w/ extra-index-urls)
# ref: https://github.com/pypa/pip/issues/4195
```

(continues on next page)

(continued from previous page)

```
venv_pip_upgrade_noconf: false

# A list of constraints to be applied when building
# or installing python packages.
venv_build_constraints: []

# A list of pip constraints to be applied as global
# constraints ahead of the list in venv_build_constraints.
# This is useful for global pins across all venvs.
venv_build_global_constraints: []

# Arguments to pass to pip when building the wheels
venv_pip_build_args: ""

# Default arguments to pass to pip when installing into
# the venv.
venv_default_pip_install_args: >-
  {%- if (groups['repo_all'] is defined) and (groups['repo_all'] | length >␣
↪0) and (venv_wheel_build_enable | bool) %}
  --find-links {{ openstack_repo_url | default('http://localhost') }}/os-
↪releases/{{ openstack_release | default('master') }}/{{ _venv_build_dist_
↪arch }}/wheels
  --trusted-host {{ (openstack_repo_url | default('http://localhost')) |␣
↪urlsplit('hostname') }}
  {%- endif %}

# Arguments to pass to pip when installing into the venv
venv_pip_install_args: ""

# Some python packages have C bindings which tend to be very
# particular about the version of their underlying shared libraries.
# To ensure things run smoothly for stable releases, we opt to
# use the distro packages for these python packages and symlink the
# appropriate python library files and their bindings into the venv.
# This variable should contain the list of packages installed which
# should be symlinked into the venv.
venv_packages_to_symlink: []

# The python executable to use for creating the venv
venv_python_executable: "python3"

# Enable the recreation of the venv from scratch.
# This is useful if you think the venv may be corrupted
# or if you have changed options which means that packages
# should be removed from the venv.
# Under normal circumstances, the installs will be done
# into the existing venv over the top of any previously
# installed packages.
venv_rebuild: no
```

(continues on next page)

(continued from previous page)

```
# Enable/disable the build of python wheels
# If the package concerned is built from a tarball, rather
# than from a git source or pypi, then this may be best to
# set to false.
venv_wheel_build_enable: "{{ venv_build_host != inventory_hostname }}"

# Set the host where the wheels will be built.
# If this host is not the same as the target host, then
# python wheels will be built in order to speed up the
# subsequent venv builds on this host and others. When
# this is the same as the target host, then we will not
# bother building wheels.
venv_build_host: "{{ venv_build_targets[ansible_facts['distribution_version
↪']][ansible_facts['architecture']] }}"

# The owner of directories and files held on the build host.
# venv_build_host_user_name: "root"
# venv_build_host_group_name: "root"

# The path for the wheel build venv.
# This is the path where a venv will be created on the
# build host for the purpose of building the wheels.
venv_build_host_venv_path: "/openstack/venvs/wheel-builder-{{ venv_python_
↪executable }}"

# The path where the requirements/constraints are stored
# on the build host in order to ensure the build process
# is idempotent.
venv_build_host_requirements_path: >-
  /var/www/repo/os-releases/{{ openstack_release | default('master') }}/{{ _
↪venv_build_dist_arch }}/requirements

# The path where the wheels are cached on the build host
# for speeding up the build process.
# TODO(odyssey4me):
# Once we remove the repo build process, this needs to point
# to the location where wheels are served by pypiserver.
# For now we use this to ensure that we don't rebuild the wheels
# that were already built by the repo build process. It has also
# been found that pypiserver hangs when it encounters duplicated
# wheels.
venv_build_host_wheel_path: >-
  /var/www/repo/os-releases/{{ openstack_release | default('master') }}/{{ _
↪venv_build_dist_arch }}/wheels

# The facts to set when the venv changes during a
# build, or the installation of a venv.
# Eg:
```

(continues on next page)

(continued from previous page)

```
# set_facts_when_changed:
#   - section: glance
#     option: venv_tag
#     value: "{{ glance_venv_tag }}"
venv_facts_when_changed: []

# The INI file name to use for the fact setting.
venv_facts_dest: "openstack_ansible"
```

1.4 References

Documentation for the project can be found at: https://docs.openstack.org/ansible-role-python_venv_build/latest/

The project home is at: https://launchpad.net/openstack-ansible

Release notes for the project can be found at: https://docs.openstack.org/releasenotes/ansible-role-python_venv_build/

The project source code repository is located at: https://git.openstack.org/cgit/openstack/ansible-role-python_venv_build

The bug tracker can be found at: https://bugs.launchpad.net/openstack-ansible