mockchroot Documentation

Release 0

Barak Korren

April 19, 2016

Contents

mock_chroot is Python library for using *mock(1)* (the chroot-based build tool, not the mockup library)

Contents:

CHAPTER 1

mock_chroot tutorial

mock_chroot lets you create *mock(1)* based chroot environemnts and run operations inside them. *mock_chroot* supports running generic shell commands as well as some higher-level operations supported by mock such as building an *RPM* package.

1.1 Basic usage

To create a *chroot* environemnt you need to simply create a *MockChroot* object:

```
from mock_chroot import MockChroot

mc = MockChroot(root='epel-6-x86_64')
```

The `root` argument is equivalent to the `--root` option of the *mock(1)* command, it takes the name of a pre-packaged chroot configuration file or a path to a custom configuration file.

Once you have a *MockChroot* object you can use it to perform operations inside the *chroot* envoronment:

```
output = mc.chroot('cat', '/etc/issue')
```

1.2 Custom configuration

The power of the *mock_chroot* module lies with the ability it provides to customize the *mock* environment from code. The most basic use is to create the configuration from a string:

```
# We read the configuration from a file here to avoid including an
# unwieldy configuration string in the example
with open('/etc/mock/epel-7-x86_64.cfg') as f:
    custom_cfg = f.read()

mc = MockChroot(config=custom_cfg)
```

Just reading configuration from files is not very interesting, so *mock_chroot* includes the *config* module which allows for programatically creating configuration snippets and composing them together:

```
import mock_chroot.config

mc = MockChroot(config=mock_chroot.config.compose(
    custom_cfg,
    mock_chroot.config.bind_mount(
```

```
        ('/dir/outside/chroot', '/dir/inside/chroot')
    )
))
```

The *config* module supports configuring various aspects of the *chroot* enviromnet including bind-mounts into it, creating files, setting environment variables and setting up network connectivity.

We can also perform more fine-grained configuration using the *mock_chroot.config.to* function:

```
mc = MockChroot(config=mock_chroot.config.compose(
    custom_cfg,
    mock_chroot.config.to['resultdir'].set(out_dir),
    mock_chroot.config.to['root_cache_enable'].set(True),
    mock_chroot.config.to['yum_cache_enable'].set(True)
))
```

1.3 Koji-based configuration

The (koji) build system can generate *mock(1)* configuration to allow one to imitate the build environments it creates. We can leverage this functionality using the *mock_chroot.config.from_koji* function:

```
mc = MockChroot(config=mock_chroot.config.from_koji(
    tag='epel7-build',
    koji_profile='koji',
))
```

The function can be combined with other *config* finctions to further customize the configuration:

```
mc = MockChroot(config=mock_chroot.config.compose(
    mock_chroot.config.from_koji(tag='epel7-build', koji_profile='koji'),
    mock_chroot.config.to['resultdir'].set(out_dir),
))
```

CHAPTER 2

mock_chroot Package reference

2.1 mock_chroot Package

mock_chroot - Thin Python wrapper around mock(1)

class mock_chroot.**MockChroot**(*root=None*, *config=None*)

Create a Python wrapper object to magae *mock(1)* chroot environments

Parameters

- **root** (*str*) – The name or path for the mock configuration file to use
- **config** (*str*) – The Mock configuration for the chroot as string or some other object that will yield a configuration string when passed to 'str()'

'root' and 'config' are mutually exclusive

dump_config()

Dump configuration for debugging purposes

Returns the current mock configuration as a string

Return type str

get_root_path()

Get the bash path of the chroot

Returns the bash path

Return type str

chroot(**cmd*, ***more_options*)

Run a non-interactive command in mock

All positional arguments are passes as the command to run and its argumens This method will behave in a simillar manner to subprocess.check_output yeilding CalledProcessError on command failure

Optional named agruments passed via 'more_options' can be as follows: :param str cwd: Working directory inside the chroot to run in

Returns the command output as string

Return type str

clean()

Clean the mock chroot

rebuild(*src_rpm*, *no_clean=False*, *define=None*, *resultdir=None*)
Build a package from .src.rpm in Mock

Parameters

- **src_rpm** (*str*) – The path to the .src.rpm file to build
- **no_clean** (*bool*) – Avoid cleaning the chroot before building
- **define** (*object*) – An optional define string for the build process or an Iterable of multiple such define strings.
- **resultdir** (*str*) – Override where the build results get placed

Returns the command output as string

Return type str

buildsrpm(*spec*, *sources*, *no_clean=False*, *define=None*, *resultdir=None*)
Build a .src.rpm package from sources and spcefile in Mock

Parameters

- **spec** (*str*) – The path to the specfile to build
- **sources** (*str*) – The path to the sources directory
- **no_clean** (*bool*) – Avoid cleaning the chroot before building
- **define** (*object*) – An optional define string for the build process or an Iterable of multiple such define strings.
- **resultdir** (*str*) – Override where the build results get placed

Returns the command output as string

Return type str

static mock_exe()
Returns the full path to the Mock executable

classmethod has_dnf()
Returns true if dnf is installed

CHAPTER 3

mock_chroot.config Package reference

3.1 mock_chroot.config Package

mock_chroot.config - Library of ways to generate Mock configuration

mock_chroot.config.**compose**([*config_objects...*])
Compose configuration objects together to form *mock(1)* chroot configuration.

Parameters **config_objects** (*list*) – Configuration objects can be strings, objects that have the *__str__()* method, or anything that is returned from one of the mock_chroot.config functions.

Returns An object representing the unified configuration. The object is an instance of a *list* subclass, so methods could be called on it to further add configuration objects. Calling *__str__()* on that object will return the composed configuration string. The object could be passed as an argument to sobsequent calls to *compose()*.

mock_chroot.config.**bind_mount**([*pairs...*])
Generate *mock(1)* bind mount configuration.

Parameters **pairs** (*list*) – List of two-element tuples where the fisrt element is a path on the host and the secont element is the path within the chroot where that path will be bind mounted

Returns A configuration object representing the bind mount configuration

mock_chroot.config.**from_koji**(*tag=None*, *target=None*, *arch='x86_64'*, *koji_profile='brew'*)
Create a koji-based *mock(1)* configuration

Parameters

- **tag** (*str*) – The Koji tag to pull configuration from
- **target** (*str*) – The Koji build target tag to pull configuration from
- **arch** (*str*) – The Koji build architecture
- **koji_profile** (*str*) – The koji configuration profile to use

One and only one of 'tag' or 'target' must be specified

Returns A configuration object containing the requested configuration

mock_chroot.config.**file**(*path*, *content*)
Add a file with given content to the mock environemnt

Parameters

- **path** (*str*) – The path to the file inside the Mock environment

- **content** (*str*) – The content of the file

Returns Mock configuration object

mock_chroot.config.**env_vars**(**vars*)
Setup environment variables inside Mock

Parameters **vars** (*dict*) – A dictionary of variables mapped to values

Returns Mock configuration object

mock_chroot.config.**use_host_resolv**()
Setup Mock to use name resolution from host

Returns Mock configuration object

m

B

C

D

E

F

G

H

M

R

U