

---

# **Molecule Documentation**

*Release 3.0a5.dev23+gd98464b4*

**AUTHORS.rst**

**Dec 09, 2019**



# CONTENTS

<b>1</b>	<b>About Ansible Molecule</b>	<b>3</b>
<b>2</b>	<b>Installation and Upgrade</b>	<b>5</b>
<b>3</b>	<b>Using Molecule</b>	<b>9</b>
<b>4</b>	<b>Common Molecule Use Cases</b>	<b>39</b>
<b>5</b>	<b>Contributing to Molecule</b>	<b>47</b>
<b>6</b>	<b>Extending Molecule</b>	<b>57</b>
<b>7</b>	<b>References and Appendices</b>	<b>59</b>
<b>8</b>	<b>Roadmaps</b>	<b>61</b>
	<b>Index</b>	<b>89</b>







## ABOUT ANSIBLE MOLECULE

Molecule is designed to aid in the development and testing of [Ansible](#) roles.

Molecule provides support for testing with multiple instances, operating systems and distributions, virtualization providers, test frameworks and testing scenarios.

Molecule encourages an approach that results in consistently developed roles that are well-written, easily understood and maintained.





## INSTALLATION AND UPGRADE

### 2.1 Installation

This document assumes the developer has a basic understanding of python packaging, and how to install and manage python on the system executing Molecule.

#### 2.1.1 Requirements

Depending on the driver chosen, you may need to install additional OS packages. See `INSTALL.rst`, which is created when initializing a new scenario.

- Python 2.7 or Python  $\geq 3.5$  with Ansible  $\geq 2.5$

##### CentOS 7

```
$ sudo yum install -y epel-release
$ sudo yum install -y gcc python-pip python-devel openssl-devel libselinux-python
```

##### Ubuntu 16.x

```
$ sudo apt-get update
$ sudo apt-get install -y python-pip libssl-dev
```

#### 2.1.2 Pip

`pip` is the only supported installation method.

Keep in mind that on selinux supporting systems, if you install into a virtual environment, you may face `issue` even if selinux is not enabled or is configured to be permissive.

It is your responsibility to assure that soft dependencies of Ansible are available on your controller or host machines.

**Warning:** It is highly recommended that you install molecule in a `virtual environment`. This will provide a modern copy of `setuptools` which is mandatory in order for molecule to be installed successfully and function correctly. If you cannot use a virtual environment then you can attempt a package upgrade with the following:

```
$ pip install --upgrade --user setuptools
```

**Warning:** Pip v19 series has an [isolation bug](#) of `setuptools` being exposed to the package build env. That is why it's highly recommended to upgrade user `setuptools` even when using a proper `virtualenv` as shown above.

## Requirements

Depending on the driver chosen, you may need to install additional python packages. See the driver's documentation or `INSTALL.rst`, which is created when initializing a new scenario.

## Install

Install Molecule:

```
$ pip install --user molecule
```

Installing molecule package also installed its main script `molecule`, usually in `PATH`. Users should know that molecule can also be called as a python module, using `python -m molecule ...`. This alternative method has some benefits:

- allows to explicitly control which python interpreter is used by molecule
- allows molecule installation at user level without even needing to have the script in `PATH`.

---

**Note:** We also have a continuous pre-release process which is provided for early adoption and feedback purposes only. It is available from [test.pypi.org/project/molecule](https://test.pypi.org/project/molecule) and can be installed like so:

```
pip install \
  --index-url https://test.pypi.org/simple \
  --extra-index-url https://pypi.org/simple \
  molecule==2.21.dev46
```

Where `2.21.dev46` is the latest available pre-release version. Please check the [release history](#) listing for the available releases.

---

### 2.1.3 Docker

We publish molecule images via [quay.io](https://quay.io) where the following tags are available:

- `latest`: latest master branch build, which should be viewed as unstable
- `2.20`: Git based tags
- `2.20a1`: Pre-releases tags

Please see the [tags listing](#) for available tags.

Please see *Docker* for usage.

### 2.1.4 Source

Due to the rapid pace of development on this tool, you might want to install and update a bleeding-edge version of Molecule from Git.

Follow the instructions below to do the initial install and subsequent updates.

The package distribution that you'll get installed will be autogenerated and will contain a commit hash information making it easier to refer to certain unstable version should the need to send a bug report arise.

**Warning:** Please avoid using `--editable/-e` [development mode](#) when installing Molecule with Pip. This not very well supported and only needed when doing development. For contributing purposes, you can rely on the `tox` command line interface. Please see [our testing guide](#) for further details.

## Requirements

### CentOS 7

```
$ sudo yum install -y libffi-devel git
```

### Ubuntu 16.x

```
$ sudo apt-get install -y libffi-dev git
```

## Install

```
$ pip install -U git+https://github.com/ansible/molecule
```

## 2.2 Upgrade Guide

TODO.



## USING MOLECULE

### 3.1 Quick Start Guide

TODO.

### 3.2 Getting Started Guide

If you're looking to move a little faster, see the *Quick Start Guide*.

The following guide will step through an example of developing and testing a new Ansible role. After reading this guide, you should be familiar with the basics of how to use Molecule and what it can offer.

#### Contents

- *Getting Started Guide*
  - *Creating a new role*
  - *Molecule Scenarios*
  - *The Scenario Layout*
  - *Inspecting the `molecule.yml`*
  - *Run test sequence commands*
  - *Run a full test sequence*

---

**Note:** In order to complete this guide by hand, you will need to additionally install [Docker](#). Molecule requires an external Python dependency for the Docker driver which is provided when installing Molecule using `pip install 'molecule[docker]'`.

---

#### 3.2.1 Creating a new role

Molecule uses [galaxy](#) under the hood to generate conventional role layouts. If you've ever worked with Ansible roles before, you'll be right at home. If not, please review the [Roles](#) guide to see what each folder is responsible for.

To generate a new role with Molecule, simply run:

```
$ molecule init role -r my-new-role
```

You should then see a `my-new-role` folder in your current directory.

---

**Note:** For future reference, if you want to initialize Molecule within an existing role, you would use the `molecule init scenario -r my-role-name` command.

---

## 3.2.2 Molecule Scenarios

You will notice one new folder which is the `molecule` folder.

In this folder is a single *Scenario* called `default`.

Scenarios are the starting point for a lot of powerful functionality that Molecule offers. For now, we can think of a scenario as a test suite for your newly created role. You can have as many scenarios as you like and Molecule will run one after the other.

## 3.2.3 The Scenario Layout

Within the `molecule/default` folder, we find a number of files and directories:

```
$ ls
Dockerfile.j2  INSTALL.rst  molecule.yml  playbook.yml  tests
```

- Since `Docker` is the default *Driver*, we find a `Dockerfile.j2` Jinja2 template file in place. Molecule will use this file to build a docker image to test your role against.
- `INSTALL.rst` contains instructions on what additional software or setup steps you will need to take in order to allow Molecule to successfully interface with the driver.
- `molecule.yml` is the central configuration endpoint for Molecule. With this file, you can configure each tool that Molecule will employ when testing your role.
- `playbook.yml` is the playbook file that contains the call site for your role. Molecule will invoke this playbook with `ansible-playbook` and run it against an instance created by the driver.
- `tests` is the tests directory created because Molecule uses `TestInfra` as the default *Verifier*. This allows you to write specific tests against the state of the container after your role has finished executing. Other verifier tools are available.

## 3.2.4 Inspecting the `molecule.yml`

The `molecule.yml` is for configuring Molecule. It is a `YAML` file whose keys represent the high level components that Molecule provides. These are:

- The *Dependency* manager. Molecule uses `std:doc:galaxy <galaxy/dev_guide>` by default to resolve your role dependencies.
- The *Driver* provider. Molecule uses `Docker` by default. Molecule uses the driver to delegate the task of creating instances.
- The *Lint* provider. Molecule uses `Yamllint` by default to ensure that best practices are encouraged when writing `YAML`.

- The *Platforms* definitions. Molecule relies on this to know which instances to create, name and to which group each instance belongs. If you need to test your role against multiple popular distributions (CentOS, Fedora, Debian), you can specify that in this section.
- The *Provisioner*. Molecule only provides an Ansible provisioner. Ansible manages the life cycle of the instance based on this configuration.
- The *Scenario* definition. Molecule relies on this configuration to control the scenario sequence order.
- The *Verifier* framework. Molecule uses `TestInfra` by default to provide a way to write specific state checking tests (such as deployment smoke tests) on the target instance.

### 3.2.5 Run test sequence commands

Let's create the first Molecule managed instance with the Docker driver.

First, ensure that `Docker` is running with the typical sanity check:

```
$ docker run hello-world
```

Now, we can tell Molecule to create an instance with:

```
$ molecule create
```

We can verify that Molecule has created the instance and they're up and running with:

```
$ molecule list
```

Now, let's add a task to our `tasks/main.yml` like so:

```
- name: Molecule Hello World!  
  debug:  
    msg: Hello, World!
```

We can then tell Molecule to test our role against our instance with:

```
$ molecule converge
```

If we want to manually inspect the instance afterwards, we can run:

```
$ molecule login
```

We now have a free hand to experiment with the instance state.

Finally, we can exit the instance and destroy it with:

```
$ molecule destroy
```

---

**Note:** If Molecule reports any errors, it can be useful to pass the `--debug` option to get more verbose output.

---

### 3.2.6 Run a full test sequence

Molecule provides commands for manually managing the lifecycle of the instance, scenario, development and testing tools. However, we can also tell Molecule to manage this automatically within a *Scenario* sequence.

The full lifecycle sequence can be invoked with:

```
$ molecule test
```

---

**Note:** It can be particularly useful to pass the `--destroy=never` flag when invoking `molecule test` so that you can tell Molecule to run the full sequence but not destroy the instance if one step fails.

---

## 3.3 Command Line Reference

### 3.3.1 Check

**class** `molecule.command.check.Check`

```
molecule check
    Target the default scenario.

molecule check --scenario-name foo
    Targeting a specific scenario.

molecule --debug check
    Executing with debug.

molecule --base-config base.yml check
    Executing with a base-config.

molecule --env-file foo.yml check
    Load an env file to read variables from when rendering molecule.yml.

molecule --parallel check
    Run in parallelizable mode.
```

### 3.3.2 Clean Up

**class** `molecule.command.cleanup.Cleanup`

This action has cleanup and is not enabled by default. See the provisioner's documentation for further details.

```
molecule cleanup
    Target the default scenario.

molecule cleanup --scenario-name foo
    Targeting a specific scenario.

molecule --debug cleanup
    Executing with debug.

molecule --base-config base.yml cleanup
    Executing with a base-config.

molecule --env-file foo.yml cleanup
    Load an env file to read variables when rendering molecule.yml.
```

### 3.3.3 Converge

Converge will execute the sequence necessary to converge the instances.



**class** molecule.command.converge.**Converge**

**molecule** converge

Target the default scenario.

**molecule** converge --scenario-name foo

Targeting a specific scenario.

**molecule** converge -- -vvv --tags foo,bar

Providing additional command line arguments to the *ansible-playbook* command. Use this option with care, as there is no sanitation or validation of input. Options passed on the CLI override options provided in provisioner's *options* section of *molecule.yml*.

**molecule** --debug converge

Executing with *debug*.

**molecule** --base-config base.yml converge

Executing with a *base-config*.

**molecule** --env-file foo.yml converge

Load an env file to read variables from when rendering molecule.yml.

### 3.3.4 Create

**class** molecule.command.create.**Create**

**molecule** create

Target the default scenario.

**molecule** create --scenario-name foo

Targeting a specific scenario.

**molecule** create --driver-name foo

Targeting a specific driver.

**molecule** --debug create

Executing with *debug*.

**molecule** --base-config base.yml create

Executing with a *base-config*.

**molecule** --env-file foo.yml create

Load an env file to read variables from when rendering molecule.yml.

### 3.3.5 Dependency

**class** molecule.command.dependency.**Dependency**

**molecule** dependency

Target the default scenario.

**molecule** dependency --scenario-name foo

Targeting a specific scenario.

**molecule** --debug dependency

Executing with *debug*.

**molecule** --base-config base.yml dependency  
Executing with a *base-config*.

**molecule** --env-file foo.yml dependency  
Load an env file to read variables from when rendering molecule.yml.

### 3.3.6 Destroy

**class** molecule.command.destroy.**Destroy**

**molecule** destroy  
Target the default scenario.

**molecule** destroy --scenario-name foo  
Targeting a specific scenario.

**molecule** destroy --all  
Target all scenarios.

**molecule** destroy --driver-name foo  
Targeting a specific driver.

**molecule** --debug destroy  
Executing with *debug*.

**molecule** --base-config base.yml destroy  
Executing with a *base-config*.

**molecule** --env-file foo.yml destroy  
Load an env file to read variables from when rendering molecule.yml.

**molecule** --parallel destroy  
Run in parallelizable mode.

### 3.3.7 Idempotence

**class** molecule.command.idempotence.**Idempotence**

Runs the converge step a second time. If no tasks will be marked as changed the scenario will be considered idempotent.

**molecule** idempotence  
Target the default scenario.

**molecule** idempotence --scenario-name foo  
Targeting a specific scenario.

**molecule** --debug idempotence  
Executing with *debug*.

**molecule** --base-config base.yml idempotence  
Executing with a *base-config*.

**molecule** --env-file foo.yml idempotence  
Load an env file to read variables from when rendering molecule.yml.

### 3.3.8 Init

**class** molecule.command.init.role.**Role**

**molecule** init role --role-name foo  
Initialize a new role.

**molecule** init role --role-name foo --template path  
Initialize a new role using ansible-galaxy and include default molecule directory. Please refer to the `init scenario` command in order to generate a custom molecule scenario.

**class** molecule.command.init.scenario.**Scenario**

**molecule** init scenario --scenario-name bar --role-name foo  
Initialize a new scenario. In order to customise the role, please refer to the *init role* command.

**cd** foo; molecule init scenario --scenario-name bar --role-name foo  
Initialize an existing role with Molecule:

**cd** foo; molecule init scenario --scenario-name bar --role-name foo --driver-template p  
Initialize a new scenario using a local *cookiecutter* template for the driver configuration.

**class** molecule.command.init.template.**Template**

**molecule** init template --url https://example.com/user/cookiecutter-repo  
Initialize a new role from a Cookiecutter URL.

### 3.3.9 Lint

**class** molecule.command.lint.**Lint**

**molecule** lint  
Target the default scenario.

**molecule** lint --scenario-name foo  
Targeting a specific scenario.

**molecule** --debug lint  
Executing with *debug*.

**molecule** --base-config base.yml lint  
Executing with a *base-config*.

**molecule** --env-file foo.yml lint  
Load an env file to read variables from when rendering molecule.yml.

### 3.3.10 List

**class** molecule.command.list.**List**

**molecule** list  
Target the default scenario.

**molecule** list --scenario-name foo  
Targeting a specific scenario.

```
molecule list --format plain
    Machine readable plain text output.

molecule list --format yaml
    Machine readable yaml output.

molecule --debug list
    Executing with debug.

molecule --base-config base.yml list
    Executing with a base-config.

molecule --env-file foo.yml list
    Load an env file to read variables from when rendering molecule.yml.
```

### 3.3.11 Login

```
class molecule.command.login.Login
```

```
molecule login
    Target the default scenario.

molecule login --scenario-name foo
    Targeting a specific scenario.

molecule login --host hostname
    Targeting a specific running host.

molecule login --host hostname --scenario-name foo
    Targeting a specific running host and scenario.

molecule --debug login
    Executing with debug.

molecule --base-config base.yml login
    Executing with a base-config.

molecule --env-file foo.yml login
    Load an env file to read variables from when rendering molecule.yml.
```

### 3.3.12 Matrix

Matrix will display the subcommand's ordered list of actions, which can be changed in [scenario](#) configuration.

```
class molecule.command.matrix.Matrix
```

```
molecule matrix subcommand
    Target the default scenario.

molecule matrix --scenario-name foo subcommand
    Targeting a specific scenario.

molecule --debug matrix subcommand
    Executing with debug.

molecule --base-config base.yml matrix subcommand
    Executing with a base-config.
```

```
molecule --env-file foo.yml matrix subcommand
    Load an env file to read variables from when rendering molecule.yml.
```

### 3.3.13 Prepare

**class** `molecule.command.prepare.Prepare`

This action is for the purpose of preparing a molecule managed instance before the `molecule.command.converge.Converge` action is run. Tasks contained within the `prepare.yml` playbook in the scenario directory will be run remotely on the managed instance. This action is run only once per test sequence.

```
molecule prepare
    Target the default scenario.

molecule prepare --scenario-name foo
    Targeting a specific scenario.

molecule prepare --driver-name foo
    Targeting a specific driver.

molecule prepare --force
    Force the execution fo the prepare playbook.

molecule --debug prepare
    Executing with debug.

molecule --base-config base.yml prepare
    Executing with a base-config.

molecule --env-file foo.yml prepare
    Load an env file to read variables from when rendering molecule.yml.
```

### 3.3.14 Side Effect

**class** `molecule.command.side_effect.SideEffect`

This action has side effects and not enabled by default. See the provisioners documentation for further details.

```
molecule side-effect
    Target the default scenario.

molecule side-effect --scenario-name foo
    Targeting a specific scenario.

molecule --debug side-effect
    Executing with debug.

molecule --base-config base.yml side-effect
    Executing with a base-config.

molecule --env-file foo.yml side-effect
    Load an env file to read variables from when rendering molecule.yml.
```

### 3.3.15 Syntax

**class** `molecule.command.syntax.Syntax`

```
molecule syntax
    Target the default scenario.
```

```
molecule syntax --scenario-name foo
    Targeting a specific scenario.

molecule --debug syntax
    Executing with debug.

molecule --base-config base.yml syntax
    Executing with a base-config.

molecule --env-file foo.yml syntax
    Load an env file to read variables from when rendering molecule.yml.
```

### 3.3.16 Test

Test will execute the sequence necessary to test the instances.

```
class molecule.command.test.Test
```

```
molecule test
    Target the default scenario.

molecule test --scenario-name foo
    Targeting a specific scenario.

molecule test --all
    Target all scenarios.

molecule test --destroy=always
    Always destroy instances at the conclusion of a Molecule run.

molecule --debug test
    Executing with debug.

molecule --base-config base.yml test
    Executing with a base-config.

molecule --env-file foo.yml test
    Load an env file to read variables from when rendering molecule.yml.

molecule --parallel test
    Run in parallelizable mode.
```

### 3.3.17 Verify

```
class molecule.command.verify.Verify
```

```
molecule verify
    Target the default scenario.

molecule verify --scenario-name foo
    Targeting a specific scenario.

molecule --debug verify
    Executing with debug.

molecule --base-config base.yml verify
    Executing with a base-config.
```

```
molecule --env-file foo.yml verify
```

Load an env file to read variables from when rendering molecule.yml.

## 3.4 Configuration

### class molecule.config.Config

Molecule searches the current directory for molecule.yml files by globbing *molecule/\*/molecule.yml*. The files are instantiated into a list of Molecule *Config* objects, and each Molecule subcommand operates on this list.

The directory in which the molecule.yml resides is the Scenario's directory. Molecule performs most functions within this directory.

The *Config* object instantiates *Dependency*, *Driver*, *Lint*, *Platforms*, *Provisioner*, *Verifier*, *Scenario*, and *State* references.

### 3.4.1 Variable Substitution

#### class molecule.interpolation.Interpolator

Configuration options may contain environment variables. For example, suppose the shell contains `VERIFIER_NAME=testinfra` and the following molecule.yml is supplied.

```
verifier:
  - name: ${VERIFIER_NAME}
```

Molecule will substitute `$VERIFIER_NAME` with the value of the `VERIFIER_NAME` environment variable.

**Warning:** If an environment variable is not set, Molecule substitutes with an empty string.

Both `$VARIABLE` and `${VARIABLE}` syntax are supported. Extended shell-style features, such as `${VARIABLE-default}` and `${VARIABLE:-default}` are also supported. Even the default as another environment variable is supported like `${VARIABLE-$DEFAULT}` or `${VARIABLE:-$DEFAULT}`. An empty string is returned when both variables are undefined.

If a literal dollar sign is needed in a configuration, use a double dollar sign (`$$`).

Molecule will substitute special `MOLECULE_` environment variables defined in *molecule.yml*.

**Important:** Remember, the `MOLECULE_` namespace is reserved for Molecule. Do not prefix your own variables with `MOLECULE_`.

A file may be placed in the root of the project as *env.yml*, and Molecule will read variables when rendering *molecule.yml*. See command usage.

### 3.4.2 Dependency

Testing roles may rely upon additional dependencies. Molecule handles managing these dependencies by invoking configurable dependency managers.

## Ansible Galaxy

**class** molecule.dependency.ansible\_galaxy.**AnsibleGalaxy**

Galaxy is the default dependency manager.

Additional options can be passed to `ansible-galaxy install` through the options dict. Any option set in this section will override the defaults.

---

**Note:** Molecule will remove any options matching `^[v]+$`, and pass `-vvv` to the underlying `ansible-galaxy` command when executing `molecule -debug`.

---

```
dependency:
  name: galaxy
  options:
    ignore-certs: True
    ignore-errors: True
    role-file: requirements.yml
```

The dependency manager can be disabled by setting `enabled` to `False`.

```
dependency:
  name: galaxy
  enabled: False
```

Environment variables can be passed to the dependency.

```
dependency:
  name: galaxy
  env:
    FOO: bar
```

## Gilt

**class** molecule.dependency.gilt.**Gilt**

Gilt is an alternate dependency manager.

Additional options can be passed to `gilt overlay` through the options dict. Any option set in this section will override the defaults.

```
dependency:
  name: gilt
  options:
    debug: True
```

The dependency manager can be disabled by setting `enabled` to `False`.

```
dependency:
  name: gilt
  enabled: False
```

Environment variables can be passed to the dependency.

```
dependency:
  name: gilt
```

(continues on next page)



(continued from previous page)

```
env:
  FOO: bar
```

## Shell

### **class** molecule.dependency.shell.Shell

Shell is an alternate dependency manager. It is intended to run a command in situations where *Ansible Galaxy* and *Gilt* don't suffice.

The `command` to execute is required, and is relative to Molecule's project directory when referencing a script not in `$PATH`.

---

**Note:** Unlike the other dependency managers, `options` are ignored and not passed to *shell*. Additional flags/subcommands should simply be added to the *command*.

---

```
dependency:
  name: shell
  command: path/to/command --flag1 subcommand --flag2
```

The dependency manager can be disabled by setting `enabled` to `False`.

```
dependency:
  name: shell
  command: path/to/command --flag1 subcommand --flag2
  enabled: False
```

Environment variables can be passed to the dependency.

```
dependency:
  name: shell
  command: path/to/command --flag1 subcommand --flag2
  env:
    FOO: bar
```

### 3.4.3 Driver

Molecule uses *Ansible* to manage instances to operate on. Molecule supports any provider *Ansible* supports. This work is offloaded to the *provisioner*.

The driver's name is specified in *molecule.yml*, and can be overridden on the command line. Molecule will remember the last successful driver used, and

continue to use the driver for all subsequent subcommands, or until the instances are destroyed by Molecule.

---

**Important:** The verifier must support the Ansible provider for proper Molecule integration.

The driver's python package requires installation.

---

## Delegated

### `class molecule.driver.delegated.Delegated`

The class responsible for managing delegated instances. Delegated is *not* the default driver used in Molecule.

Under this driver, it is the developers responsibility to implement the create and destroy playbooks. Managed is the default behaviour of all drivers.

```
driver:
  name: delegated
```

However, the developer must adhere to the instance-config API. The developer's create playbook must provide the following instance-config data, and the developer's destroy playbook must reset the instance-config.

```
- address: ssh_endpoint
  identity_file: ssh_identity_file # mutually exclusive with password
  instance: instance_name
  port: ssh_port_as_string
  user: ssh_user
  password: ssh_password # mutually exclusive with identity_file
  become_method: valid_ansible_become_method # optional
  become_pass: password_if_required # optional

- address: winrm_endpoint
  instance: instance_name
  connection: 'winrm'
  port: winrm_port_as_string
  user: winrm_user
  password: winrm_password
  winrm_transport: ntlm/credssp/kerberos
  winrm_cert_pem: <path to the credssp public certificate key>
  winrm_cert_key_pem: <path to the credssp private certificate key>
  winrm_server_cert_validation: validate/ignore
```

This article covers how to configure and use WinRM with Ansible: [https://docs.ansible.com/ansible/latest/user\\_guide/windows\\_winrm.html](https://docs.ansible.com/ansible/latest/user_guide/windows_winrm.html)

Molecule can also skip the provisioning/deprovisioning steps. It is the developers responsibility to manage the instances, and properly configure Molecule to connect to said instances.

```
driver:
  name: delegated
  options:
    managed: False
    login_cmd_template: 'docker exec -ti {instance} bash'
    ansible_connection_options:
      ansible_connection: docker
platforms:
  - name: instance-docker
```

```
$ docker run \
  -d \
  --name instance-docker \
  --hostname instance-docker \
  -it molecule_local/ubuntu:latest sleep infinity & wait
```

Use Molecule with delegated instances, which are accessible over ssh.

---

**Important:** It is the developer's responsibility to configure the ssh config file.

---

```
driver:
  name: delegated
  options:
    managed: False
    login_cmd_template: 'ssh {instance} -F /tmp/ssh-config'
    ansible_connection_options:
      ansible_connection: ssh
      ansible_ssh_common_args: '-F /path/to/ssh-config'
platforms:
  - name: instance
```

Provide the files Molecule will preserve post destroy action.

```
driver:
  name: delegated
  safe_files:
    - foo
```

And in order to use localhost as molecule's target:

```
driver:
  name: delegated
  options:
    managed: False
    ansible_connection_options:
      ansible_connection: local
```

## Docker

### `class molecule.driver.docker.Docker`

The class responsible for managing [Docker](#) containers. [Docker](#) is the default driver used in Molecule.

Molecule leverages Ansible's [docker\\_container](#) module, by mapping variables from `molecule.yml` into `create.yml` and `destroy.yml`.

```
driver:
  name: docker
platforms:
  - name: instance
    hostname: instance
    image: image_name:tag
    dockerfile: Dockerfile.j2
    pull: True|False
    pre_build_image: True|False
    registry:
      url: registry.example.com
      credentials:
        username: $USERNAME
        password: $PASSWORD
        email: user@example.com
        user: root
    override_command: True|False
```

(continues on next page)

(continued from previous page)

```

command: sleep infinity
tty: True|False
pid_mode: host
privileged: True|False
security_opts:
  - seccomp=unconfined
devices:
  - /dev/fuse:/dev/fuse:rwm
volumes:
  - /sys/fs/cgroup:/sys/fs/cgroup:ro
keep_volumes: True|False
tmpfs:
  - /tmp
  - /run
capabilities:
  - SYS_ADMIN
sysctls:
  net.core.somaxconn: 1024
  net.ipv4.tcp_syncookies: 0
exposed_ports:
  - 53/udp
  - 53/tcp
published_ports:
  - 0.0.0.0:8053:53/udp
  - 0.0.0.0:8053:53/tcp
ulimits:
  - nofile:262144:262144
dns_servers:
  - 8.8.8.8
etc_hosts: '{"host1.example.com': '10.3.1.5'}"
networks:
  - name: foo
  - name: bar
network_mode: host
purge_networks: true
docker_host: tcp://localhost:12376
cacert_path: /foo/bar/ca.pem
cert_path: /foo/bar/cert.pem
key_path: /foo/bar/key.pem
tls_verify: true
env:
  FOO: bar
restart_policy: on-failure
restart_retries: 1
buildargs:
  http_proxy: http://proxy.example.com:8080/

```

If specifying the `CMD` directive in your `Dockerfile.j2` or consuming a built image which declares a `CMD` directive, then you must set `override_command: False`. Otherwise, Molecule takes care to honour the value of the `command` key or uses the default of `bash -c "while true; do sleep 10000; done"` to run the container until it is provisioned.

When attempting to utilize a container image with `systemd` as your init system inside the container to simulate a real machine, make sure to set the `privileged`, `volumes`, `command`, and `environment` values. An example using the `centos:7` image is below:

---

**Note:** Do note that running containers in privileged mode is considerably less secure. For details, please reference [Docker Security Configuration](#)

---

**Note:** With the environment variable `DOCKER_HOST` the user can bind Molecule to a different [Docker](#) socket than the default `unix:///var/run/docker.sock`. `tcp`, `fd` and `ssh` socket types can be configured. For details, please reference [Docker daemon socket options](#).

---

```
platforms:
- name: instance
  image: centos:7
  privileged: true
  volumes:
  - "/sys/fs/cgroup:/sys/fs/cgroup:rw"
  command: "/usr/sbin/init"
  tty: True
  environment:
  container: docker
```

```
$ pip install molecule[docker]
```

When pulling from a private registry, it is the user's discretion to decide whether to use hard-code strings or environment variables for passing credentials to molecule.

---

**Important:** Hard-coded credentials in `molecule.yml` should be avoided, instead use *variable substitution*.

---

Provide a list of files Molecule will preserve, relative to the scenario ephemeral directory, after any `destroy` subcommand execution.

```
driver:
  name: docker
  safe_files:
  - foo
```

## Podman

**class** `molecule.driver.podman.Podman`

The class responsible for managing [Podman](#) containers. [Podman](#) is not default driver used in Molecule.

Molecule uses Podman ansible connector and podman CLI while mapping variables from `molecule.yml` into `create.yml` and `destroy.yml`.

```
driver:
  name: podman
platforms:
- name: instance
  hostname: instance
  image: image_name:tag
  dockerfile: Dockerfile.j2
  pull: True|False
  pre_build_image: True|False
```

(continues on next page)

(continued from previous page)

```

registry:
  url: registry.example.com
  credentials:
    username: $USERNAME
    password: $PASSWORD
  override_command: True|False
  command: sleep infinity
  tty: True|False
  pid_mode: host
  privileged: True|False
  security_opts:
    - seccomp=unconfined
  volumes:
    - /sys/fs/cgroup:/sys/fs/cgroup:ro
  tmpfs:
    - /tmp
    - /run
  capabilities:
    - SYS_ADMIN
  exposed_ports:
    - 53/udp
    - 53/tcp
  published_ports:
    - 0.0.0.0:8053:53/udp
    - 0.0.0.0:8053:53/tcp
  ulimits:
    - nofile=1024:1028
  dns_servers:
    - 8.8.8.8
  network: host
  etc_hosts: {'host1.example.com': '10.3.1.5'}
  cert_path: /foo/bar/cert.pem
  tls_verify: true
  env:
    FOO: bar
  restart_policy: on-failure
  restart_retries: 1
  buildargs:
    http_proxy: http://proxy.example.com:8080/

```

If specifying the `CMD` directive in your `Dockerfile.j2` or consuming a built image which declares a `CMD` directive, then you must set `override_command: False`. Otherwise, Molecule takes care to honour the value of the `command` key or uses the default of `bash -c "while true; do sleep 10000; done"` to run the container until it is provisioned.

When attempting to utilize a container image with `systemd` as your init system inside the container to simulate a real machine, make sure to set the `privileged`, `volumes`, `command`, and `environment` values. An example using the `centos:7` image is below:

---

**Note:** Do note that running containers in privileged mode is considerably less secure.

---

```

platforms:
- name: instance
  image: centos:7
  privileged: true

```

(continues on next page)

(continued from previous page)

```
command: "/usr/sbin/init"
tty: True
```

```
$ pip install molecule[podman]
```

When pulling from a private registry, it is the user's discretion to decide whether to use hard-code strings or environment variables for passing credentials to molecule.

---

**Important:** Hard-coded credentials in `molecule.yml` should be avoided, instead use *variable substitution*.

---

Provide a list of files Molecule will preserve, relative to the scenario ephemeral directory, after any `destroy` subcommand execution.

```
driver:
  name: podman
  safe_files:
    - foo
```

### 3.4.4 Lint

Molecule handles project linting by invoking configurable linters.

#### Yaml Lint

**class** `molecule.lint.yamllint.Yamllint`

`yamllint` is the default project linter.

`yamllint` is a linter for YAML files. In addition to checking for syntax validity it also checks for key repetition as well as cosmetic problems such as line length, trailing spaces, and indentation.

The default `yamllint` settings that ship with `molecule` can be found in the [cookiecutter template](#).

Additional options can be passed to `yamllint` through the `options` dict. Any option set in this section will override the defaults. See the [yamllint rules](#) for more options.

```
lint:
  name: yamllint
  options:
    config-file: foo/bar
```

The project linting can be disabled by setting `enabled` to `False`.

```
lint:
  name: yamllint
  enabled: False
```

Environment variables can be passed to lint.

```
lint:
  name: yamllint
  env:
    FOO: bar
```

Paths can be ignored.

```
lint:
  name: yamllint
  options:
    config-data:
      ignore: path_to_ignore
```

### 3.4.5 Platforms

**class** `molecule.platforms.Platforms`

Platforms define the instances to be tested, and the groups to which the instances belong.

```
platforms:
  - name: instance-1
```

Multiple instances can be provided.

```
platforms:
  - name: instance-1
  - name: instance-2
```

Mapping instances to groups. These groups will be used by the *Provisioner* for orchestration purposes.

```
platforms:
  - name: instance-1
    groups:
      - group1
      - group2
```

Children allow the creation of groups of groups.

```
platforms:
  - name: instance-1
    groups:
      - group1
      - group2
    children:
      - child_group1
```

### 3.4.6 Provisioner

Molecule handles provisioning and converging the role.

#### Ansible

**class** `molecule.provisioner.ansible.Ansible`

*Ansible* is the default provisioner. No other provisioner will be supported.

Molecule's provisioner manages the instances lifecycle. However, the user must provide the create, destroy, and converge playbooks. Molecule's `init` subcommand will provide the necessary files for convenience.

Molecule will skip tasks which are tagged with either *molecule-notest* or *notest*. With the tag *molecule-idempotence-notest* tasks are only skipped during the idempotence action step.



---

**Important:** Reserve the create and destroy playbooks for provisioning. Do not attempt to gather facts or perform operations on the provisioned nodes inside these playbooks. Due to the gymnastics necessary to sync state between Ansible and Molecule, it is best to perform these tasks in the prepare or converge playbooks.

It is the developers responsibility to properly map the modules's fact data into the `instance_conf_dict` fact in the create playbook. This allows Molecule to properly configure Ansible inventory.

---

Additional options can be passed to `ansible-playbook` through the `options` dict. Any option set in this section will override the defaults.

---

**Important:** Options do not affect the create and destroy actions.

---

**Note:** Molecule will remove any options matching `^[v]+$`, and pass `-vvv` to the underlying `ansible-playbook` command when executing `molecule -debug`.

---

Molecule will silence log output, unless invoked with the `--debug` flag. However, this results in quite a bit of output. To enable Ansible log output, add the following to the `provisioner` section of `molecule.yml`.

```
provisioner:
  name: ansible
  log: True
```

The create/destroy playbooks for Docker and Podman are bundled with Molecule. These playbooks have a clean API from `molecule.yml`, and are the most commonly used. The bundled playbooks can still be overridden.

The playbook loading order is:

1. `provisioner.playbooks.$driver_name.$action`
2. `provisioner.playbooks.$action`
3. `bundled_playbook.$driver_name.$action`

```
provisioner:
  name: ansible
  options:
    vvv: True
  playbooks:
    create: create.yml
    converge: playbook.yml
    destroy: destroy.yml
```

Share playbooks between roles.

```
provisioner:
  name: ansible
  playbooks:
    create: ../default/create.yml
    destroy: ../default/destroy.yml
    converge: playbook.yml
```

Multiple driver playbooks. In some situations a developer may choose to test the same role against different backends. Molecule will choose driver specific create/destroy playbooks, if the determined driver has a key in the `playbooks` section of the `provisioner's` dict.

**Important:** If the determined driver has a key in the playbooks dict, Molecule will use this dict to resolve all provisioning playbooks (create/destroy).

---

```
provisioner:
  name: ansible
  playbooks:
    docker:
      create: create.yml
      destroy: destroy.yml
    create: create.yml
    destroy: destroy.yml
    converge: playbook.yml
```

**Important:** Paths in this section are converted to absolute paths, where the relative parent is the `$scenario_directory`.

---

The side effect playbook executes actions which produce side effects to the instances(s). Intended to test HA failover scenarios or the like. It is not enabled by default. Add the following to the provisioner's `playbooks` section to enable.

```
provisioner:
  name: ansible
  playbooks:
    side_effect: side_effect.yml
```

**Important:** This feature should be considered experimental.

---

The prepare playbook executes actions which bring the system to a given state prior to converge. It is executed after create, and only once for the duration of the instances life.

This can be used to bring instances into a particular state, prior to testing.

```
provisioner:
  name: ansible
  playbooks:
    prepare: prepare.yml
```

The cleanup playbook is for cleaning up test infrastructure that may not be present on the instance that will be destroyed. The primary use-case is for “cleaning up” changes that were made outside of Molecule’s test environment. For example, remote database connections or user accounts. Intended to be used in conjunction with *prepare* to modify external resources when required.

The cleanup step is executed directly before every destroy step. Just like the destroy step, it will be run twice. An initial clean before converge and then a clean before the last destroy step. This means that the cleanup playbook must handle failures to cleanup resources which have not been created yet.

Add the following to the provisioner’s *playbooks* section to enable.

```
provisioner:
  name: ansible
  playbooks:
    cleanup: cleanup.yml
```

---

**Important:** This feature should be considered experimental.

---

Environment variables. Molecule does its best to handle common Ansible paths. The defaults are as follows.

```
ANSIBLE_ROLES_PATH:
  $ephemeral_directory/roles/:$project_directory/../../~/.ansible/roles:/usr/share/
↔ansible/roles:/etc/ansible/roles
ANSIBLE_LIBRARY:
  $ephemeral_directory/modules/:$project_directory/library/~/~/.ansible/plugins/
↔modules:/usr/share/ansible/plugins/modules
ANSIBLE_FILTER_PLUGINS:
  $ephemeral_directory/plugins/filter/:$project_directory/filter/plugins/~/~/.
↔ansible/plugins/filter:/usr/share/ansible/plugins/modules
```

Environment variables can be passed to the provisioner. Variables in this section which match the names above will be appended to the above defaults, and converted to absolute paths, where the relative parent is the `$scenario_directory`.

---

**Important:** Paths in this section are converted to absolute paths, where the relative parent is the `$scenario_directory`.

---

```
provisioner:
  name: ansible
  env:
    FOO: bar
```

Modifying `ansible.cfg`.

```
provisioner:
  name: ansible
  config_options:
    defaults:
      fact_caching: jsonfile
    ssh_connection:
      scp_if_ssh: True
```

---

**Important:** The following keys are disallowed to prevent Molecule from improperly functioning. They can be specified through the provisioner's `env` setting described above, with the exception of the `privilege_escalation`.

---

```
provisioner:
  name: ansible
  config_options:
    defaults:
      roles_path: /path/to/roles_path
      library: /path/to/library
      filter_plugins: /path/to/filter_plugins
      privilege_escalation: {}
```

Roles which require host/groups to have certain variables set. Molecule uses the same variables defined in a playbook syntax as Ansible.

```
provisioner:
  name: ansible
  inventory:
    group_vars:
      foo1:
        foo: bar
      foo2:
        foo: bar
        baz:
          qux: zzyzx
    host_vars:
      foo1-01:
        foo: bar
```

Molecule automatically generates the inventory based on the hosts defined under *Platforms*. Using the `hosts` key allows to add extra hosts to the inventory that are not managed by Molecule.

A typical use case is if you want to access some variables from another host in the inventory (using `hostvars`) without creating it.

---

**Note:** The content of `hosts` should follow the YAML based inventory syntax: start with the `all` group and have `hosts/vars/children` entries.

---

```
provisioner:
  name: ansible
  inventory:
    hosts:
      all:
        extra_host:
          foo: hello
```

---

**Important:** The extra hosts added to the inventory using this key won't be created/destroyed by Molecule. It is the developers responsibility to target the proper hosts in the playbook. Only the hosts defined under *Platforms* should be targetted instead of `all`.

---

An alternative to the above is symlinking. Molecule creates symlinks to the specified directory in the inventory directory. This allows ansible to converge utilizing its built in `host/group_vars` resolution. These two forms of inventory management are mutually exclusive.

Like above, it is possible to pass an additional inventory file (or even dynamic inventory script), using the `hosts` key. Ansible will automatically merge this inventory with the one generated by molecule. This can be useful if you want to define extra hosts that are not managed by Molecule.

---

**Important:** Again, it is the developers responsibility to target the proper hosts in the playbook. Only the hosts defined under *Platforms* should be targetted instead of `all`.

---

---

**Note:** The source directory linking is relative to the scenario's directory.

The only valid keys are `hosts`, `group_vars` and `host_vars`. Molecule's schema validator will enforce this.

---

```

provisioner:
  name: ansible
  inventory:
    links:
      hosts: ../../../../inventory/hosts
      group_vars: ../../../../inventory/group_vars/
      host_vars: ../../../../inventory/host_vars/

```

Override connection options:

```

provisioner:
  name: ansible
  connection_options:
    ansible_ssh_user: foo
    ansible_ssh_common_args: -o IdentitiesOnly=no

```

Add arguments to ansible-playbook when running converge:

```

provisioner:
  name: ansible
  ansible_args:
    - --inventory=mygroups.yml
    - --limit=host1,host2

```

## Lint

Molecule handles provisioner linting by invoking configurable linters.

**class** `molecule.provisioner.lint.ansible_lint.AnsibleLint`

*Ansible Lint* is the default role linter.

*Ansible Lint* checks playbooks for practices, and behaviour that could potentially be improved.

Additional options can be passed to *ansible-lint* through the options dict. Any option set in this section will override the defaults.

```

provisioner:
  name: ansible
  lint:
    name: ansible-lint
    options:
      exclude:
        - path/exclude1
        - path/exclude2
      x: ["ANSIBLE0011,ANSIBLE0012"]
      force-color: True

```

The *x* option has to be passed like this due to a [bug](#) in Ansible Lint.

The role linting can be disabled by setting `enabled` to `False`.

```

provisioner:
  name: ansible
  lint:
    name: ansible-lint
    enabled: False

```

Environment variables can be passed to lint.

```
provisioner:
  name: ansible
  lint:
    name: ansible-lint
    env:
      FOO: bar
```

### 3.4.7 Scenario

Molecule treats scenarios as a first-class citizens, with a top-level configuration syntax.

#### **class** molecule.scenario.Scenario

A scenario allows Molecule test a role in a particular way, this is a fundamental change from Molecule v1.

A scenario is a self-contained directory containing everything necessary for testing the role in a particular way. The default scenario is named `default`, and every role should contain a default scenario.

Unless mentioned explicitly, the scenario name will be the directory name hosting the files.

Any option set in this section will override the defaults.

```
scenario:
  name: default # optional
  create_sequence:
    - dependency
    - create
    - prepare
  check_sequence:
    - dependency
    - cleanup
    - destroy
    - create
    - prepare
    - converge
    - check
    - destroy
  converge_sequence:
    - dependency
    - create
    - prepare
    - converge
  destroy_sequence:
    - dependency
    - cleanup
    - destroy
  test_sequence:
    - dependency
    - lint
    - cleanup
    - destroy
    - syntax
    - create
    - prepare
    - converge
    - idempotence
    - side_effect
```

(continues on next page)

(continued from previous page)

```
- verify
- cleanup
- destroy
```

### 3.4.8 State

An internal bookkeeping mechanism.

#### **class** `molecule.state.State`

A class which manages the state file. Intended to be used as a singleton throughout a given Molecule config. The initial state is serialized to disk if the file does not exist, otherwise is deserialized from the existing state file. Changes made to the object are immediately serialized.

State is not a top level option in Molecule's config. It's purpose is for bookkeeping, and each *Config* object has a reference to a *State* object.

---

**Note:** Currently, it's use is significantly smaller than it was in v1 of Molecule.

---

### 3.4.9 Verifier

Molecule handles role testing by invoking configurable verifiers.

#### **Ansible**

#### **class** `molecule.verifier.ansible.Ansible`

*Ansible* is not the default test runner.

Molecule executes a playbook (*verify.yml*) located in the role's *scenario.directory*.

```
verifier:
  name: ansible
  lint:
    name: ansible-lint
```

The testing can be disabled by setting `enabled` to `False`.

```
verifier:
  name: ansible
  enabled: False
```

Environment variables can be passed to the verifier.

```
verifier:
  name: ansible
  env:
    FOO: bar
```

## Lint

**class** molecule.verifier.lint.ansible\_lint.**AnsibleLint**

`AnsibleLint` is not the default verifier linter.

`AnsibleLint` checks playbooks for practices, and behaviour that could potentially be improved.

Additional options can be passed to `ansible-lint` through the options dict. Any option set in this section will override the defaults.

```
verifier:
  name: ansible
  lint:
    name: ansible-lint
    options:
      exclude:
        - path/exclude1
        - path/exclude2
      x: ["ANSIBLE0011,ANSIBLE0012"]
      force-color: True
```

The `x` option must be passed like this due to a [bug](#) in Ansible Lint.

The role linting can be disabled by setting `enabled` to `False`.

```
verifier:
  name: ansible
  lint:
    name: ansible-lint
    enabled: False
```

Environment variables can be passed to lint.

```
verifier:
  name: ansible
  lint:
    name: ansible-lint
    env:
      FOO: bar
```

## Testinfra

**class** molecule.verifier.testinfra.**Testinfra**

`Testinfra` is the default test runner.

Additional options can be passed to `testinfra` through the options dict. Any option set in this section will override the defaults.

---

**Note:** Molecule will remove any options matching `^[v]+[extract_itex]`, and pass `-vvv` to the underlying `pytest` command when executing `molecule --debug`.

---

```
verifier:
  name: testinfra
  options:
    n: 1
```



The testing can be disabled by setting `enabled` to `False`.

```
verifier:
  name: testinfra
  enabled: False
```

Environment variables can be passed to the verifier.

```
verifier:
  name: testinfra
  env:
    FOO: bar
```

Change path to the test directory.

```
verifier:
  name: testinfra
  directory: /foo/bar/
```

Additional tests from another file or directory relative to the scenario's tests directory (supports regexp).

```
verifier:
  name: testinfra
  additional_files_or_dirs:
    - ../path/to/test_1.py
    - ../path/to/test_2.py
    - ../path/to/directory/*
```

## Lint

**class** `molecule.verifier.lint.flake8.Flake8`

`Flake8` is the default verifier linter.

`Flake8` is a linter for python files.

Additional options can be passed to `flake8` through the options dict. Any option set in this section will override the defaults.

```
verifier:
  name: testinfra
  lint:
    name: flake8
    options:
      benchmark: True
```

Test file linting can be disabled by setting `enabled` to `False`.

```
verifier:
  name: testinfra
  lint:
    name: flake8
    enabled: False
```

Environment variables can be passed to lint.

```
verifier:  
  name: testinfra  
  lint:  
    name: flake8  
  env:  
    FOO: bar
```

**class** molecule.verifier.lint.precommit.PreCommit

Pre-commit tool verifier wrapper.

This class is used to lint files by executing the pre-commit command line tool for files in the test folder with a prefix of test\_.

Pre-Commit is not the default verifier linter.

Pre-Commit is a linter for python files and more.

Additional options can be passed to pre-commit through the options dict. Any option set in this section will override the defaults.

```
verifier:  
  name: testinfra  
  lint:  
    name: pre-commit  
    options:  
      remove-tabs:
```

Test file linting can be disabled by setting enabled to False.

```
verifier:  
  name: testinfra  
  lint:  
    name: pre-commit  
    enabled: False
```

Environment variables can be passed to lint.

```
verifier:  
  name: testinfra  
  lint:  
    name: pre-commit  
  env:  
    FOO: bar
```

Example pre-commit configuration file (.pre-commit-config.yaml) to run flake8.

```
repos:  
  - repo: local  
    hooks:  
      - id: flake8  
        name: flake8  
        entry: python -m flake8 --max-line-length=120  
        language: system  
        types: [python]
```

## COMMON MOLECULE USE CASES

### 4.1 Common Molecule Use Cases

#### 4.1.1 Docker

Molecule can be executed via an Alpine Linux container by leveraging dind (Docker in Docker). Currently, we only build images for the latest version of Ansible and Molecule. In the future we may break this out into Molecule/ Ansible versioned pairs. The images are located on [quay.io](https://quay.io).

To test a role, change directory into the role to test, and execute Molecule as follows.

```
docker run --rm -it \  
  -v "$(pwd)":/tmp/$(basename "${PWD}") :ro \  
  -v /var/run/docker.sock:/var/run/docker.sock \  
  -w /tmp/$(basename "${PWD}") \  
  quay.io/ansible/molecule:2.20 \  
  molecule test
```

#### 4.1.2 Docker With Non-Privileged User

The default Molecule Docker driver executes Ansible playbooks as the root user. If your workflow requires a non-privileged user, then adapt `molecule.yml` and `Dockerfile.j2` as follows.

Append the following code block to the end of `Dockerfile.j2`. It creates an `ansible` user with passwordless sudo privileges.

The variable `SUDO_GROUP` depends on the target distribution. `centos:7` uses `wheel`.

```
# Create `ansible` user with sudo permissions and membership in `DEPLOY_GROUP`  
ENV ANSIBLE_USER=ansible SUDO_GROUP=wheel DEPLOY_GROUP=deployer  
RUN set -xe \  
  && groupadd -r ${ANSIBLE_USER} \  
  && groupadd -r ${DEPLOY_GROUP} \  
  && useradd -m -g ${ANSIBLE_USER} ${ANSIBLE_USER} \  
  && usermod -aG ${SUDO_GROUP} ${ANSIBLE_USER} \  
  && usermod -aG ${DEPLOY_GROUP} ${ANSIBLE_USER} \  
  && sed -i "/^%${SUDO_GROUP}/s/ALL\$/NOPASSWD:ALL/g" /etc/sudoers
```

Modify `provisioner.inventory` in `molecule.yml` as follows:

```
platforms:  
  - name: instance
```

(continues on next page)

(continued from previous page)

```
image: centos:7
# ...
```

```
provisioner:
  name: ansible
  # ...
  inventory:
    host_vars:
      # setting for the platform instance named 'instance'
      instance:
        ansible_user: ansible
```

Make sure to use your **platform instance name**. In this case `instance`.

An example for a different platform instance name:

```
platforms:
- name: centos7
  image: centos:7
  # ...
```

```
provisioner:
  name: ansible
  # ...
  inventory:
    host_vars:
      # setting for the platform instance named 'centos7'
      centos7:
        ansible_user: ansible
```

To test it, add the following task to `tasks/main.yml`. It fails, because the non-privileged user is not allowed to create a folder in `/opt/`. This needs to be performed using `sudo`.

To perform the task using `sudo`, uncomment `become: yes`. Now the task will succeed.

```
- name: Create apps dir
  file:
    path: /opt/examples
    owner: ansible
    group: deployer
    mode: 0775
    state: directory
  # become: yes
```

Don't forget to run `molecule destroy` if image has already been created.

### 4.1.3 Systemd Container

To start a service which requires `systemd`, in a non-privileged container, configure `molecule.yml` with a `systemd` compliant image, `tmpfs`, `volumes`, and `command` as follows.

```
platforms:
- name: instance
  image: centos:7
  command: /sbin/init
```

(continues on next page)

(continued from previous page)

```

tmpfs:
  - /run
  - /tmp
volumes:
  - /sys/fs/cgroup:/sys/fs/cgroup:ro

```

Note that centos:7 image contains a [seccomp security profile for Docker](#) which enables the use of systemd. When needed, such security profiles can be reused (for example [the one available in Fedora](#)):

```

platforms:
  - name: instance
    image: debian:stretch
    command: /sbin/init
    security_opts:
      - seccomp=path/to/seccomp.json
    tmpfs:
      - /run
      - /tmp
    volumes:
      - /sys/fs/cgroup:/sys/fs/cgroup:ro

```

The developer can also opt to [start the container with extended privileges](#), by either giving it SYS\_ADMIN capabilities or running it in privileged mode.

---

**Important:** Use caution when using privileged mode or SYS\_ADMIN capabilities as it grants the container elevated access to the underlying system.

---

To limit the scope of the extended privileges, grant SYS\_ADMIN capabilities along with the same image, command, and volumes as shown in the non-privileged example.

```

platforms:
  - name: instance
    image: centos:7
    command: /sbin/init
    capabilities:
      - SYS_ADMIN
    volumes:
      - /sys/fs/cgroup:/sys/fs/cgroup:ro

```

To start the container in privileged mode, set the privileged flag along with the same image and command as shown in the non-privileged example.

```

platforms:
  - name: instance
    image: centos:7
    command: /sbin/init
    privileged: True

```

#### 4.1.4 Monolith Repo

Molecule is generally used to test roles in isolation. However, it can also test roles from a monolith repo.

```
$ tree monolith-repo -L 3 --prune
monolith-repo
├── library
│   └── foo.py
├── plugins
│   └── filters
│       └── foo.py
└── roles
    ├── bar
    │   └── README.md
    ├── baz
    │   └── README.md
    └── foo
        └── README.md
```

The role initialized with Molecule (baz in this case) would simply reference the dependant roles via it's `playbook.yml` or meta dependencies.

Molecule can test complex scenarios leveraging this technique.

```
$ cd monolith-repo/roles/baz
$ molecule test
```

Molecule is simply setting the `ANSIBLE_*` environment variables. To view the environment variables set during a Molecule operation pass the `--debug` flag.

```
$ molecule --debug test

DEBUG: ANSIBLE ENVIRONMENT
---
ANSIBLE_CONFIG: /private/tmp/monolith-repo/roles/baz/molecule/default/.molecule/
↪ansible.cfg
ANSIBLE_FILTER_PLUGINS: /Users/jodewey/.pyenv/versions/2.7.13/lib/python2.7/site-
↪packages/molecule/provisioner/ansible/plugins/filters:/private/tmp/monolith-repo/
↪roles/baz/plugins/filters:/private/tmp/monolith-repo/roles/baz/molecule/default/.
↪molecule/plugins/filters
ANSIBLE_LIBRARY: /Users/jodewey/.pyenv/versions/2.7.13/lib/python2.7/site-packages/
↪molecule/provisioner/ansible/plugins/libraries:/private/tmp/monolith-repo/roles/baz/
↪library:/private/tmp/monolith-repo/roles/baz/molecule/default/.molecule/library
ANSIBLE_ROLES_PATH: /private/tmp/monolith-repo/roles:/private/tmp/monolith-repo/roles/
↪baz/molecule/default/.molecule/roles
```

Molecule can be customized any number of ways. Updating the provisioner's `env` section in `molecule.yml` to suit the needs of the developer and layout of the project.

```
provisioner:
  name: ansible
  env:
    ANSIBLE_${VAR}: $VALUE
```

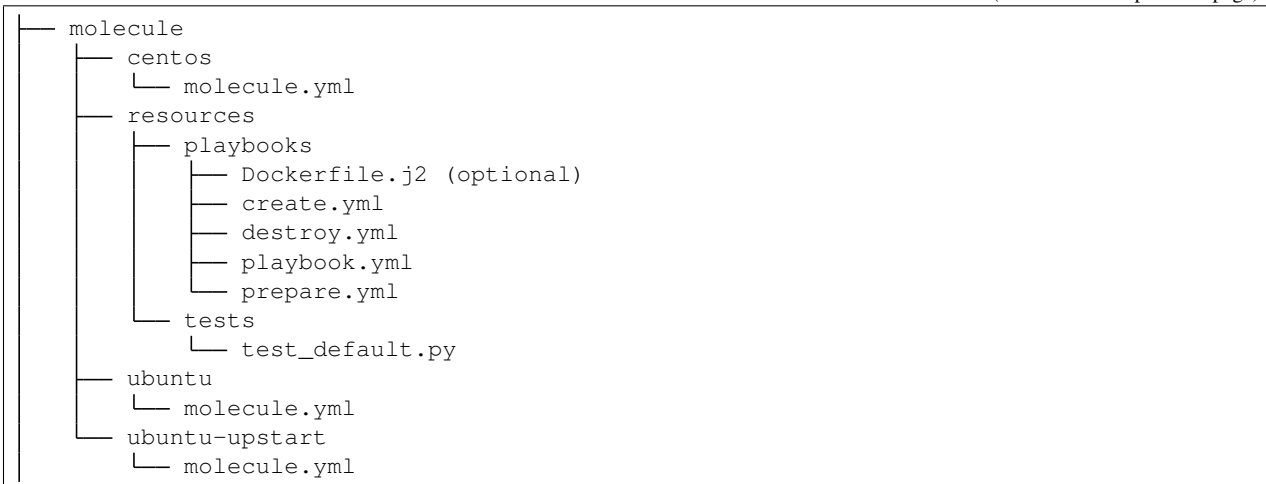
## 4.1.5 Sharing Across Scenarios

Playbooks and tests can be shared across scenarios.

```
$ tree shared-tests
shared-tests
```

(continues on next page)

(continued from previous page)



Tests can be shared across scenarios. In this example the `tests` directory lives in a shared location and `molecule.yml` is points to the shared tests.

```

verifier:
name: testinfra
directory: ../resources/tests/
lint:
  name: flake8

```

#### 4.1.6 Running Molecule processes in parallel mode

---

**Important:** This functionality should be considered experimental. It is part of ongoing work towards enabling parallelizable functionality across all moving parts in the execution of the Molecule feature set.

---

**Note:** Only the following sequences support parallelizable functionality:

- `check_sequence:molecule check --parallel`
- `destroy_sequence:molecule destroy --parallel`
- `test_sequence:molecule test --parallel`

It is currently only available for use with the Docker driver.

---

It is possible to run Molecule processes in parallel using another tool to orchestrate the parallelization (such as [GNU Parallel](#) or [Pytest](#)).

When Molecule receives the `--parallel` flag it will generate a **UUID** for the duration of the testing sequence and will use that unique identifier to cache the run-time state for that process. The parallel Molecule processes cached state and created instances will therefore not interfere with each other.

Molecule uses a new and separate caching folder for this in the `$HOME/.cache/molecule_parallel` location. Molecule exposes a new environment variable `MOLECULE_PARALLEL` which can enable this functionality.

## 4.2 FAQ

### 4.2.1 Why is my idempotence action failing?

It is important to understand that Molecule does not do anything further than the default functionality of Ansible when determining if your tasks are idempotent or not. Molecule will simply run the converge action twice and check against Ansible's standard output.

Therefore, if you are seeing idempotence failures, it is typically related to the underlying Ansible report and not Molecule.

If you are facing idempotence failures and intend to raise a bug on our issue tracker, please first manually run `molecule converge` twice and confirm that Ansible itself is reporting task idempotence (`changed=0`).

### 4.2.2 Why does Molecule make so many shell calls?

Ansible provides a Python API. However, it is not intended for [direct consumption](#). We wanted to focus on making Molecule useful, so our efforts were spent consuming Ansible's CLI.

Since we already consume Ansible's CLI, we decided to call additional binaries through their respective CLI.

---

**Note:** This decision may be reevaluated later.

---

### 4.2.3 Why does Molecule only support Ansible versions 2.2 and later?

- Ansible 2.2 is the first good release in the Ansible 2 lineup.
- The modules needed to support the drivers did not exist pre 2.2 or were not sufficient.

### 4.2.4 Why are playbooks used to provision instances?

Simplicity. Ansible already supports numerous cloud providers. Too much time was spent in Molecule v1, re-implementing a feature that already existed in the core Ansible modules.

### 4.2.5 Have you thought about using Ansible's python API instead of playbooks?

This was [evaluated](#) early on. It was a toss up. It would provide simplicity in some situations and complexity in others. Developers know and understand playbooks. Decided against a more elegant and sexy solution.

### 4.2.6 Why are there multiple scenario directories and molecule.yml files?

Again, simplicity. Rather than defining an all encompassing config file opted to normalize. Molecule simply loops through each scenario applying the scenario's molecule.yml.

---

**Note:** This decision may be reevaluated later.

---



### 4.2.7 Are there similar tools to Molecule?

- Ansible’s own Testing Strategies
- ansible-test (abandoned?)
- RoleSpec

### 4.2.8 Can I run Molecule processes in parallel?

Please see *Running Molecule processes in parallel mode* for usage.

### 4.2.9 Can I specify random instance IDs in my molecule.yml?

This depends on the CI provider but the basic recipe is as follows.

Setup your `molecule.yml` to look like this:

```
platforms:
  - name: "instance-#{INSTANCE_UUID}"
```

Then in your CI provider environment, for example, Gitlab CI, setup:

```
variables:
  INSTANCE_UUID: "$CI_JOB_ID"
```

Where `CI_JOB_ID` is the random variable that Gitlab provides.

Molecule will resolve the `INSTANCE_UUID` environment variable when creating and looking up the instance name. You can confirm all is in working order by running `molecule list`.

### 4.2.10 Can I test Ansible Collections with Molecule?

This is not currently officially supported. Also, collections remain in “tech preview” status. However, you can take a look at [this blog post](#) outlining a workable “DIY” solution as a stop gap for now.

### 4.2.11 Does Molecule support monorepos?

Yes, roles contained in a [monorepo](#) with other roles are automatically picked up and `ANSIBLE_ROLES_PATH` is set accordingly. See [this page](#) for more information.

### 4.2.12 How can I add development/testing-only dependencies?

Sometimes, it’s desirable to only run a dependency role when developing your role with molecule, but not impose a hard dependency on the role itself; for example when you rely on one of its side effects. This can be achieved by an approach like this in your role’s `meta/main.yml`:

```
---
dependencies:
  - role: <your-dependee-role>
    when: lookup('env', 'MOLECULE_FILE')
```



## CONTRIBUTING TO MOLECULE

### 5.1 Contributing

#### 5.1.1 Move to Red Hat

---

**Important:** During the end of October 2018 the Molecule Project was moved to its new home under Ansible by Red Hat.

---

##### How to get involved

- To see what's planned see the [Molecule Project Board](#).
- Join the Molecule [community working group](#) if you would like to influence the direction of the project.

##### Update Git repo location

The Molecule project has moved:

old location: `https://github.com/metacloud/molecule`

new location: `https://github.com/ansible/molecule`

If you have the source checked out you should use `git remote set-url origin` to point to the new location.

Please follow GitHub's official [changing a remote's URL](#) guide.

##### New Docker location

For people that use the Docker image, we are now publishing to a new location [Molecule on quay.io](#).

old location: `https://hub.docker.com/r/retr0h/molecule/`

new location: `https://quay.io/repository/ansible/molecule`

How to use:

```
docker pull quay.io/ansible/molecule:latest
```

## Release announcements

Want to know about releases, subscribe to [ansible-announce](#) list

## Talk to us

Join us in [#ansible-molecule](#) on [freenode](#), or [molecule-users](#) Forum.

The full list of Ansible email lists and IRC channels can be found in the [communication](#) page.

## 5.1.2 Contribution Guidelines

- We are interested in various different kinds of improvement for Molecule; please feel free to raise an [Issue](#) if you would like to work on something major to ensure efficient collaboration and avoid duplicate effort.
- Create a topic branch from where you want to base your work.
- Check for unnecessary whitespace with `git diff --check` before committing. Please see [formatting and linting](#) documentation for further commands.
- Make sure you have added tests for your changes.
- Although not required, it is good to sign off commits using `git commit --signoff`, and agree that usage of `--signoff` constitutes agreement with the terms of [DCO 1.1](#).
- Run all the tests to ensure nothing else was accidentally broken.
- Reformat the code by following the formatting section below.
- Submit a pull request.

## 5.1.3 Code Of Conduct

Please see our [Code of Conduct](#) document.

## 5.1.4 Pull Request Life Cycle and Governance

- If your PRs get stuck [join us on IRC](#) or add to the [working group](#) agenda.
- The code style is what is enforced by CI, everything else is off topic.
- All PRs must be reviewed by one other person. This is enforced by GitHub. Larger changes require +2.

## 5.1.5 Installing

*Installation* from source or package.

## 5.1.6 Testing

Please see *Full*.

## 5.1.7 Documentation

### Working with InterSphinx

In the `conf.py`, we define an `intersphinx_mapping` which provides the base URLs for conveniently linking to other Sphinx documented projects. In order to find the correct link syntax and text you can link to, you can quickly inspect the reference from the command line.

For example, if we would like to link to a specific part of the Ansible documentation, we could first run the following command:

```
python -m sphinx.ext.intersphinx https://docs.ansible.com/ansible/latest/objects.inv
```

And then see the entire Sphinx listing. We see entries that look like:

```
py:attribute
  AnsibleModule._debug  api/index.html#AnsibleModule._debug
```

With which we can link out to using the following syntax:

```
:py:attribute:`AnsibleModule._debug`
```

## 5.2 Testing

Molecule has an extensive set of unit and functional tests. Molecule uses [Tox Factors](#) to generate a matrix of python x Ansible x unit/functional tests. Manual setup required as of this time.

### 5.2.1 Dependencies

Tests will be skipped when the driver's binary is not present.

Install the test framework [Tox](#).

```
$ pip install tox
```

### 5.2.2 Full

Run all tests, including linting and coverage reports. This should be run prior to merging or submitting a pull request.

```
$ tox
```

### 5.2.3 List available scenarios

List all available scenarios. This is useful to target specific Python and Ansible version for the functional and unit tests.

```
$ tox -l
```

## 5.2.4 Unit

Run all unit tests with coverage.

```
$ tox -e 'py{27,35,36,37,38}-ansible{25,26,27,28}-unit'
```

Run all unit tests for a specific version of Python and Ansible (here Python 3.7 and Ansible 2.7).

```
$ tox -e py37-ansible28-unit
```

## 5.2.5 Linting

Linting is performed by a combination of linters.

Run all the linters (some perform changes to conform the code to the style rules).

```
$ tox -e lint
```

## 5.2.6 Documentation

Generate the documentation, using [sphinx](#).

```
$ tox -e doc
```

## 5.2.7 Build docker

Build the docker container.

```
$ tox -e build-docker
```

## 5.2.8 Continuous integration

Molecule output will use ANSI colors if stdout is an interactive TTY and TERM value seems to support it. You can define `PY_COLORS=1` to force use of ANSI colors, which can be handy for some CI systems.

### Github Actions

[GitHub Actions](#) runs a CI pipeline, much like any others, that's built into GitHub.

An action to clone a repo as `molecule_demo`, and run `molecule test` in `ubuntu`.

```
---
name: Molecule Test
on: [push, commit]
jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      max-parallel: 4
    matrix:
```

(continues on next page)

(continued from previous page)

```

python-version: [3.5, 3.6, 3.7]

steps:
- uses: actions/checkout@v1
  with:
    path: molecule_demo
- name: Set up Python ${ matrix.python-version }
  uses: actions/setup-python@v1
  with:
    python-version: ${ matrix.python-version }
- name: Install dependencies
  run: |
    sudo apt install docker
    python -m pip install --upgrade pip
    pip install -r requirements.txt
- name: Test with molecule
  run: |
    molecule test

```

## Travis CI

Travis is a CI platform, which can be used to test Ansible roles.

A `.travis.yml` testing a role named `foo1` with the Docker driver.

```

---
sudo: required
language: python
services:
- docker
before_install:
- sudo apt-get -qq update
install:
- pip install molecule
  # - pip install required driver (e.g. docker, shade, boto, apache-libcloud)
script:
- molecule test

```

A `.travis.yml` using `Tox` as described below.

```

---
sudo: required
language: python
services:
- docker
before_install:
- sudo apt-get -qq update
install:
- pip install tox-travis
script:
- tox

```

## Gitlab CI

Gitlab includes its own CI. Pipelines are usually defined in a `.gitlab-ci.yml` file in the top folder of a repository,

to be ran on Gitlab Runners.

Here is an example setting up a virtualenv and testing an Ansible role via Molecule. User-level pip is cached and so is the virtual environment to save time. And this is run over a runner tagged *pip36* and *docker*, because its a minimal CentOS 7 VM installed with pip36 from IUS repository and docker.

```
---
image: docker:git

services:
  - docker:dind

before_script:
  - apk update && apk add --no-cache docker
  - python3-dev py3-pip docker gcc git curl build-base
  - autoconf automake py3-cryptography linux-headers
  - musl-dev libffi-dev openssl-dev openssl
  - docker info
  - python3 --version

molecule:
stage: test
script:
  - pip3 install ansible molecule docker
  - ansible --version
  - cd roles/testrole && molecule test
```

## Jenkins Pipeline

Jenkins projects can also be defined in a file, by default named *Jenkinsfile* in the top folder of a repository. Two syntax are available, Declarative and Scripted. Here is an example using the declarative syntax, setting up a virtualenv and testing an Ansible role via Molecule.

```
pipeline {
  agent {
    // Node setup : minimal centos7, plugged into Jenkins, and
    // git config --global http.sslVerify false
    // sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm
    // sudo yum -y install python36u python36u-pip python36u-devel git curl gcc
    // git config --global http.sslVerify false
    // sudo curl -fsSL get.docker.com | bash
    label 'Molecule_Slave'
  }

  stages {
    stage ('Get latest code') {
      steps {
        checkout scm
      }
    }

    stage ('Setup Python virtual environment') {
      steps {
        sh '''
          export HTTP_PROXY=http://10.123.123.123:8080
        '''
      }
    }
  }
}
```

(continues on next page)



(continued from previous page)

```

        export HTTPS_PROXY=http://10.123.123.123:8080
        pip3.6 install virtualenv
        virtualenv virtenv
        source virtenv/bin/activate
        pip install --upgrade ansible molecule docker
        '''
    }
}

stage ('Display versions') {
    steps {
        sh '''
            source virtenv/bin/activate
            docker -v
            python -V
            ansible --version
            molecule --version
            '''
    }
}

stage ('Molecule test') {
    steps {
        sh '''
            source virtenv/bin/activate
            molecule test
            '''
    }
}
}
}

```

The following *Jenkinsfile* uses the official 'quay.io/ansible/molecule' image.

```

pipeline {
    agent {
        docker {
            image 'quay.io/ansible/molecule'
            args '-v /var/run/docker.sock:/var/run/docker.sock'
        }
    }

    stages {

        stage ('Display versions') {
            steps {
                sh '''
                    docker -v
                    python -V
                    ansible --version
                    molecule --version
                    '''
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```
stage ('Molecule test') {
  steps {
    sh 'sudo molecule test --all'
  }
}

} // close stages
} // close pipeline
```

**Note:** For Jenkins to work properly using a *Multibranch Pipeline* or a *GitHub Organisation* - as used by Blue Ocean, the role name in the scenario/playbook.yml should be changed to perform a lookup of the role root directory. For example :

```
---
- name: Converge
  hosts: all
  roles:
    - role: "{{ lookup('env', 'MOLECULE_PROJECT_DIRECTORY') | basename }}"
```

This is the cleaner of the current choices. See [issue1567\\_comment](#) for additional detail.

## Tox

Tox is a generic virtualenv management, and test command line tool. Tox can be used in conjunction with [Factors](#) and [Molecule](#), to perform scenario tests.

To test the role against multiple versions of Ansible.

```
[tox]
minversion = 1.8
envlist = py{27}-ansible{20,21,22}
skipdist = true

[testenv]
passenv = *
deps =
  -requirements.txt
  ansible20: ansible==2.0.2.0
  ansible21: ansible==2.1.2.0
  ansible22: ansible==2.2.0.0
commands =
  molecule test
```

To view the factor generated tox environments.

```
$ tox -l
py27-ansible20
py27-ansible21
py27-ansible22
```

If using the `--parallel` functionality of Tox (version 3.7 onwards), Molecule must be made aware of the parallel testing by setting a `MOLECULE_EPHEMERAL_DIRECTORY` environment variable per environment. In addition, we export a `TOX_ENVNAME` environment variable, it's the name of our tox env.

```

[tox]
minversion = 3.7
envlist = py{27}_ansible{23,24}
skipsdist = true

[testenv]
deps =
    -rrequirements.txt
    ansible23: ansible==2.3
    ansible24: ansible==2.4
commands =
    molecule test
setenv =
    TOX_ENVNAME={envname}
    MOLECULE_EPHEMERAL_DIRECTORY=/tmp/{envname}

```

You also must include the `TOX_ENVNAME` variable in name of each platform in `molecule.yml` configuration file. This way, their names won't create any conflict.

```

---
dependency:
  name: galaxy
driver:
  name: docker
lint:
  name: yamllint
platforms:
  - name: instance1-${TOX_ENVNAME}
    image: mariadb
  - name: instance2-${TOX_ENVNAME}
    image: retr0h/centos7-systemd-ansible:latest
    privileged: True
    command: /usr/sbin/init
provisioner:
  name: ansible
  lint:
    name: ansible-lint
verifier:
  name: testinfra
  lint:
    name: flake8

```

## 5.3 Development

- Please read the *Contributing* guidelines.

### 5.3.1 Release

Molecule follows Semantic Versioning.

#### Docker Build

- Quay.io automatically builds on commit and tag

## 5.4 Credits

Based on the good work of John Dewey (@retr0h).

### 5.4.1 Contributors

Please see the following:

- [Contributors listing](#)
- [Community working group members](#)

**EXTENDING MOLECULE**

TODO.



## REFERENCES AND APPENDICES

- genindex





## 8.1 Changelog

### 8.1.1 Unreleased

- Moved the Hetzner Cloud driver out to a plugin
- MAJOR: Removed LXC and LXD providers
- MAJOR: Dockerfile templates are now embedded in molecule.
- MINOR: Fixed typo with OVERRIDDEN placeholder in templates
- Supported Ansible versions are now 2.9, 2.8, 2.7
- Removed goss verifier
- dependency now runs before lint on default test and lint sequences
- ANSIBLE\_ROLES\_PATH, ANSIBLE\_LIBRARY, ANSIBLE\_FILTER\_PLUGINS now include the default Ansible lookup paths `/usr/share/ansible/<roles/filter/modules>` and `/etc/ansible/roles`
- The internal Molecule plugins are moved to paths more like upstream. `ansible/plugins/filters > ansible/plugins/filter` and `ansible/plugins/libraries > ansible/plugins/modules`
- Bash style variable expansion for environment variable defaults added. `foo: ${UNDEFINED_VAR:-$DEFAULT}` and `foo: ${UNDEFINED_VAR-$DEFAULT}` are now supported.
- Use pluggy to load plugins
- Windows & Linux EC2 instances can be tested simultaneously
- Windows EC2 instances can be provisioned using the automatically-generated password

### 8.1.2 2.22

- `molecule dependency` now has a retry and timed back-off by default for flaky network connections.
- Add the `-parallel` flag to experimentally allow molecule to be run in parallel.
- `dependency` step is now run by default before any playbook sequence step, including `create` and `destroy`. This allows the use of roles in all sequence step playbooks.
- Removed validation regex for docker registry passwords, all `string` values are now valid.
- Add `tty` option to the Docker driver.

- Specify new lower bound of 3.0.2 for `testinfra` which uses the new Ansible test runner.
- Place upper bounds on `inspec` and `rubocop` for CI testing.
- Support pruning of docker volumes in ‘destroy’ phase for docker driver
- Update Goss to 0.3.7
- Add SSH password to delegated driver instance\_dict
- Add WinRM connections options to delegated driver instance\_dict
- Update testinfra to 3.0.6 so we can use ansible verbosity
- Add `sysctl`s option to the Docker driver.

### 8.1.3 2.20

#### Important Changes

- Project now maintained by the Ansible Team, see [Move to Red Hat](#) for details
- Docker Container now hosted on [quay.io](#)

#### Other

- Molecule docker images will use the following convention on tags going forwards:
  - `latest`: corresponds to the master branch, which should be viewed as unstable
  - `2.20`: Git based tags
  - `2.20a1`: pre-releases tags
- Molecule docker image no longer requires `sudo` when invoking `molecule`.
- Molecule docker image no longer specifies `USER molecule`.
- Officially advertise support for Python 3.5.
- Remove mandatory `-r` option for `molecule init scenario`.
- Make the default scenario use the parent folder.
- Fix support for honouring environment variables such as `MOLECULE_DEBUG`.
- Allow to customise the location of the `Dockerfile.j2` with the `dockerfile` option for the Docker driver.
- Add integer type coercion for the `exposed_ports` platform option.
- Add support for honouring `PY_COLORS` environment variable.
- Disable YAML lint truthy rule by default.
- Add validation for non-unique platform instance names.
- Add ‘Getting Started’ guide to the documentation for the benefit of new users.
- Allow to specify extra inventory sources not created by Molecule.
- Avoid including assets in the package `sdist`.
- Add `openssh-client` to the Molecule Docker image.
- Fix `ca-certificates` installation for OpenSUSE.

- Add `purge_networks` option to the Docker driver.
- Add `pid_mode` option to the Docker driver.
- Constrain `ansible-lint` to `>=4.0.2, <5`.
- Add the Linode driver (API v3).
- Provide documented example for using `systemd` enabled Docker images.
- Add `winrm` connection support for the delegated driver.
- Remove usage of `sudo pip . .` in driver installation documentation.
- Add `override_command` option to the Docker driver for overriding CMD directives.
- Only recommend to install `'molecule[docker]'` in the `INSTALL.rst` for the Docker driver.
- Sort scenario execution order by directory name.
- Fix Python package install for Docker `prepare.yml` on Fedora Rawhide.
- Update SHA-256 hash for the Goss binary.
- Remove `Detox` (deprecated) configuration example from `Tox` documentation.
- Add `CODE_OF_CONDUCT.md`.
- Add optional `cleanup` sequence step.
- Allow to customise configuration file location with `MOLECULE_GLOB` environment variable.
- Molecule can now be called as a Python module (`python -m molecule`). Patch by [@ssbarnea](#).
- Add [Travis CI integration](#) and fix related test issues.
- Add Docker `buildargs` option for configuring the `docker_image create.yml` build step.

### 8.1.4 2.19

- Bumped `testinfra` to 1.16.0 due to `testinfra` bug.
- Allows lowercase environment variables in the Docker scheme.
- Removes local mode from LXD documentation.

### Important Changes

Last release by [@retr0h](#). Subsequent releases will be made by the Ansible team.

### 8.1.5 2.18.1

- Fixes #1484 - add `ruby-etc` apk package.
- Fix documentation of scenario sequences.

### 8.1.6 2.18

- Bump Goss to v0.3.6.
- Fixes docs build, appends #egg to tox-tags url.
- Fixes typo in base.py status docstring.
- Adds to goss docs about linting.
- Deprecated ansible 2.2 and 2.3 tests.
- Bumped ansible versions to test.
- Docs: Recommend prepare playbook for node setup.
- Updates typo in docker section of test\_platforms\_section.py.
- Adds install instructions for RuboCop.
- Updates tox-tags url in test-requirements.txt.
- Add support of restart\_policy and restart\_retries to docker driver.
- Added TERM=xterm to docker instance env.
- Added network\_mode option to Docker container.
- Adds pre\_build\_image option to Docker create playbook.
- Remove the double *init* in the doc.
- Expand LXD driver functionality.
- Fixed the matrix subcommand yet again.

### 8.1.7 2.17

- Correct .env file interpolation.
- Fixes Tox link in docs.
- Adds tox-tags to test-requirements.txt.
- Expose config.project\_directory as env var.
- Update Matrix usage.rst.
- Update ci.rst with Jenkinsfile example.
- Support passing arbitrary keys to vm.network.
- Pin wheel version to 0.30.0.
- Add git to docker DIND container.
- Added inspec download for Ubuntu 14.04.
- Added env to docker.
- Accept a single option to the matrix subcommand.
- Knob to change Ansible *no\_log*.
- Bumped testinfra to 1.14.1 due to testinfra bug.
- Remove upgrade from Dockerfile.

- Bumped requirements.txt.
- Corrected provider\_override\_args.
- Add docker python and rubocop dependencies.
- Added python 3.7 support.

### 8.1.8 2.16

- Add feature for auto bumping docker image tag.
- Fixed Docker provider not using DOCKER\_HOST environmental variable.
- Updates to the Ansible provisioning playbook for docker and vagrant for missing options.
- Documentation : dependencies on centos and docker driver clarifications.
- Added matrix subcommand.
- added pull: yeslno param to Docker executor.
- Added Gitlab CI example.
- Add information about the action which failed.
- Support Ansible 2.6.
- Corrected schema due to #1344.
- Prevalidator should enforce allowed options.
- Add support for multiple distributions to inspec verifier.
- Update InSpec to version 2.2.20.
- Update ansible-lint to version 3.4.23.
- Create unique keypair to allow parallel executions with OpenStack driver.
- Requirements update.
- Update the Dockerfile for work with az client and rubocop.

### 8.1.9 2.15

- Removed docker credential regexp validation.
- Added rsync to Docker image.
- Docker create playbooks: add tmpfs & security\_opts docker\_container parameters.
- Moved default scenario to a const.
- Pre-validate Molecule special variables.
- Added env file.
- Corrected command syntax.
- Delegated driver acts as managed.

### 8.1.10 2.14

- Add pre-validation.
- MOLECULE\_ special variables available in molecule.yml.
- Log Vagrant stdout to a file in MOLECULE\_EPHEMERAL\_DIRECTORY.
- Reintroduce base config merging.
- Corrected unit tests to work with tox.
- Add verifier mutually exclusive checking.
- UTF-8 issue in idempotence.
- Made prepare playbook optional.
- Bundle common playbooks.
- Added Goss linter.
- Disallow verifier.options with Goss and Inspec.

### Important Changes

- MOLECULE\_ special variables available in molecule.yml.
- Molecule introduces a new CLI option `-base-config`, which is loaded prior to each scenario's `molecule.yml`. This allows developers to specify a base config, to help reduce repetition in their molecule.yml files. The default base config is `~/.config/molecule/config.yml`.
- Prepare playbook no longer needs to exist, unless using it.
- Molecule bundles Docker and Vagrant create/destroy playbooks.

### 8.1.11 2.13.1

- Enable Ansible 2.4 support with py36.

### 8.1.12 2.13

- Allow the destroying of remote libvirt instances.
- Bumped testinfra version for Ansible 2.5.1 compatibility.
- Added RuboCop as Inspec's linter.
- Minor fixes to Goss verifier playbook.
- Update documentation for verify and idempotency checks.
- Added Inspec verifier.
- Support Void Linux when using Docker driver.
- Converge with built in Molecule skip tags.
- Render inventory links relative to scenario dir.
- Disallow null provider.env values.
- Log vagrant errors.

- Enable py36 support for Ansible 2.5.
- Retry downloading goss 3 times.
- Delegated driver should report unknown on *molecule list*.
- Correct Docker container terminal sizing.
- Bumped Ansible 2.4 minor version in tox.
- Add `docker_host` attribute to templates to allow talking to a remote docker daemon.
- Across-the-board requirements update.
- Add parameter for Vagrant provider override.
- Add 'halt' option to Vagrant module.

### Important Changes

- Python 3.6 support.
- Added Inspec verifier.
- Added RuboCop linter for Inspec.

### Breaking Changes

- Render inventory links relative to scenario dir instead of ephemeral dir. Unfortunately, this was a side effect of #1218.

#### 8.1.13 2.12.1

- Disable pytest caching plugin.

### Important Changes

- No longer need to *.gitignore* the *.pytest\_cache/* directory.

#### 8.1.14 2.12

- Ensure `prune` properly removes empty dirs.
- Allow `verify` playbook to be shared.
- Added `cookiecutter` tests.
- Moved temporary files to `$TMPDIR`.
- Added and tested Ansible 2.5 support.
- Remove `include` tasks from driver playbooks.
- Set `delete_fip = yes` for `os_server` resources.
- Relaxed schema validation for which allows unknown keys in *molecule.yml*.
- Corrected AnsibleLint `-x` example.
- Added `dind` support and docs.

- Exclude `.venv` directory from `yamllint`.
- Move Molecule playbook vars into host inventory.
- Switch functional tests to `pytest.raises`.

### Important Changes

- Molecule writes temporary files to `$TMPDIR` hashed as `molecule/$role_name/$scenario_name/`. Temporary files are no longer written to `$scenario_directory/molecule/`.
- No longer need to `.gitignore` the `.molecule/` directory.

### Breaking Changes

- Users of the Goss verifier will need to change their `verifier.yml` playbook to `verify.yml`.

## 8.1.15 2.11

- Correct verbose flag options with `-debug`.
- Bumped Ansible 2.4 and 2.3 minor versions.
- Reimplemented schema validation with Cerberus.
- Bumped version of `jinja2`.
- Move `merge_dicts` into `util`.
- Forward port Molecule v1 shell dependency manager.
- Vagrantfile cleanup.
- Ability to log into a Docker registry.

### Important Changes

- Reimplemented schema validation with Cerberus. The Molecule file is thoroughly validated. This may result in validation errors if the developer's `molecule.yml` is doing something unusual.
- Cleaned up the Vagrantfile, and allow the developer to change options on the base Vagrant config object.

### Breaking Changes

- Changed Vagrant's `molecule.yml raw_config_args` to `provider_raw_config_args` for differentiating `instance_raw_config_args`.

## 8.1.16 2.10.1

- Correct Vagrant to automatically insert a keypair.
- Corrected `synced_folders` usage.



### 8.1.17 2.10

- Properly skipping Vagrant speedup keys in provider.
- Allow Vagrant to automatically insert a keypair.
- Correct molecule\_vagrant.py bug where *provider\_options* would cause Vagrant to fail if keys from #1147 were provided.
- Fix line length in cookie cutter README.

#### Important Changes

- PR #1147 reduced Vagrant create time, which disabled Vagrant from automatically inserting a keypair. Molecule's default is now changed back to Vagrant's default of True, which may reduce the speed of Vagrant create as fixed by #1147.

### 8.1.18 2.9

- Bumped yamllint version.
- Namespaced Docker registry.
- Reduce create time with Vagrant driver.
- Replace >>> with \$ in documentation.
- Moved prune to run after destroy.
- Fix confusion between exposed and published ports in docker create playbook.
- Add basic support for libvirt in Vagrant driver.
- Ignore psutil on cygwin platform.
- Corrected ability to set multiple x options in provisioner's lint.
- Disallow privilege\_escalation via schema.
- Validate schema for invalid ansible config options.
- Adding provision option for Vagrant driver.

#### Important Changes

- These changes do not impact existing projects. However, if one was using the old syntax, and upgraded create.yml, changes would be required. The Docker driver's registry has been moved to a key named *url* under *registry*.

```
driver:
  name: docker
platforms:
- name: instance
  image: image_name:tag
  registry:
    url: registry.example.com
```

- Fix confusion between exposed and published ports in docker create playbook.

```
driver:
  name: docker
platforms:
- name: instance
  image: image_name:tag
  exposed_ports:
    - "53/udp"
    - "53/tcp"
  published_ports:
    - "0.0.0.0:8053:53/udp"
    - "0.0.0.0:8053:53/tcp"
```

### 8.1.19 2.8.2

- Corrected ansible args.

### 8.1.20 2.8.1

- Reverted, release does not exist.

### 8.1.21 2.8

- Improved quickstart video.
- Ability to specify a custom registry to Docker driver.
- Add a link to talk demo.
- Corrected incorrectly fixed bug when tags provided to provisioner.
- Corrected dependency scenario functional tests.
- Corrected incorrectly fixed bug when providing provisioner lint options.
- Regexp support in `additional_files_or_dirs`.
- Add custom nameserver to Docker container.
- Add network create and destroy support to Docker driver.

### Breaking Changes

- The verifier's `additional_files_or_dirs` option is relative to the test directory, as opposed to the scenario directory.
- The verifier's `additional_files_or_dirs` option now supports regexp. Molecule will add additional files or directories, only when the glob succeeds. Directories must be appended with the regexp to match, further details in the verifier's documentation.

### 8.1.22 2.7

- Ability to set a ulimit for the Docker driver.
- Switching `log_driver` from none to json-file to for compatibility with Ansible 2.2.
- Default to always destroy strategy.

- Support `linked_clone` for Vagrant 2.X.
- Bump `tree-format` to 0.1.2.
- Correct starting container on Docker edge by changing `log_driver` to `none`.
- Make `psutil` installation platform-dependent.

### 8.1.23 2.6

- Path searching to check ephemeral dir first.
- Update Goss `verifier.yml`.
- Bump `ansible-lint` version.
- Added example for setting Vagrant proxy settings for Linux.
- Never destroy instances if `--destroy-never` requested.
- Variable `Molecule Ephemeral Directory`.
- Added `systemd` example.

### 8.1.24 2.5

- Ignore `provisioner.options` when in the `create/destroy` `provisioner`.
- Switched Docker driver to a portable default command.
- Parallel instance management.
- Added Azure driver.
- Corrected `testinfra SystemInfo` tests.
- Execute `dependency` on check and converge sequence.
- Updated Docs usage of `dependency` role-file instead of `requirements_file`.
- Cleaned up YAML syntax.
- Execute linting first in test sequence.
- Support `expose_ports` option in `docker` driver.

### 8.1.25 2.4

- Corrected missing code block inside documentation.
- Bump `ansible-lint` version.
- Added `yamllint` to `init` scenario.
- Correct `env` path qualification.
- Add `sudo` package to Fedora section of `Dockerfile` template.
- Correct `ANSIBLE_ROLES_PATH` path component.
- Allow re-run of `prepare` playbook.

### 8.1.26 2.3

- Report friendly error message when interpolation fails.
- Added a new line after printing matrix.
- Added molecule header to generated Dockerfiles.
- Check supported python and ansible versions when executing Molecule.
- Sanitize user provided config options.
- Sanitize user provided env options.
- Added shell friendly env output

### 8.1.27 2.2.1

- Ensure setup is run for prepare to correct ssh connection failures.

### 8.1.28 2.2

- Ability to execute a prepare playbook post create.
- Log deprecation warning when missing prepare.yml.
- Support Ansible 2.4.
- Revert “Add support import data from original ansible.cfg”.
- Changed testinfra command to py.test.

### 8.1.29 2.1

- Add a destroy strategy to the *test* action.
- Delegated driver may or may not manage instances.

### 8.1.30 2.0.4

- Fix Dockerfile for Fedora.

### 8.1.31 2.0.3

- Generate host/group vars when host vars missing.

### 8.1.32 2.0.2

- Pass the provisioner’s env to the verifier.

### 8.1.33 2.0.1

- Corrected init scenario validation.

### 8.1.34 2.0

- Major overhaul of Molecule.

#### Important Changes

- Ansible playbooks to manage instances.
- Vagrant is managed through a custom Ansible module bundled with Molecule.
- Addition of [Scenarios](#).
- Addition of a [Delegated Driver](#) to test instances managed outside of Molecule.
- Promoted [Goss Verifier](#) to a supported verifier.
- Added [GCE Driver](#), [EC2 Driver](#), [LXC Driver](#), [LXD Driver](#) , and [OpenStack Driver](#) native Molecule drivers.

#### Breaking Changes

- Not compatible with Molecule v1 style config.
- Demoted serverspec support entirely.
- Does not support all of the Molecule v1 functionality or flexibility, in favor of simplicity and consistency throughout.
- Ansible 2.2 and 2.3 support only.
- See Molecule v1 to v2 [Porting Guide](#).
- Molecule no longer defaults to passing the `-become` flag to the `ansible-playbook` command.
- Roles are linted with [Yamllint](#) vs v1's custom linter.

### 8.1.35 1.25.1

- Update ansible-lint for Ansible 2.4 compatibility.

### 8.1.36 1.25

- Display output when *idempotence* fails.
- Changed basebox to ubuntu/trusty64 for molecule init.
- Allow `disable_cache` parameter for Docker containers enhancement.
- Update goss verifier.
- Add a 'private' parameter in OpenStack driver.

### 8.1.37 1.24

- Support Ansible 2.3.

### 8.1.38 1.23.3

- Clean up {group,host}\_vars on destroy.

### 8.1.39 1.23.2

- Globally disable cowsay, since it impacts the idempotence check.

### 8.1.40 1.23.1

- Added ungrouped hosts under all.

### 8.1.41 1.23

- Prescriptive ansible.cfg defaults.
- Ansible v2 has deprecated ansible\_ssh\_{host,port,user}.
- Docker driver: use POSIX shell and support more linux package systems.
- Add quotes around ansible\_ssh\_private\_key\_file format.
- Ansible 1.9 No longer supported.

### 8.1.42 1.22

- Handling of networks with Docker driver.

### 8.1.43 1.21.1

- Corrected None RepoTags bug with docker driver.

### 8.1.44 1.21

- No longer skip setting hostname with Vagrant's libvirt provider.
- Openstack: Allow using ssh keys from ssh-agent.
- Obtain driver from state file if set.
- Updated to Goss 0.3.0.
- Remove terminal warnings while running apt.
- Support for new docker sdk.
- Updated doc for docker driver links.

## Breaking Changes

- The *docker-py* pip package has been deprecated in favor of *docker*.

### 8.1.45 1.20.3

- Version bump, network interruption while uploading package to pypi.

### 8.1.46 1.20.2

- Correct testinfra tests discovered twice.

### 8.1.47 1.20.1

- Correct too many authentication failures error.

### 8.1.48 1.20

- Expose network configuration to docker driver.
- Openstack: Performance improvements for multiinstance setups.
- Do not require a `project_config` when a `local_config` is present.
- Corrected molecule.yml's `group_vars/host_vars`.

## Breaking Changes

- The `host_vars` and `group_vars` section of molecule.yml no longer accepts a list, rather a dict similar to Ansible's `vars` usage.

### 8.1.49 1.19.3

- Openstack: Use configured ssh key.

### 8.1.50 1.19.2

- Properly handle testinfra verbose flag setting.

### 8.1.51 1.19.1

- Add `raw_config_args` option to providers.

### 8.1.52 1.19

- Convert vagrantfile from relying on jinja.

### 8.1.53 1.18.1

- Make Openstack ssh timeout configurable.

### 8.1.54 1.18

- Fix availability timeout in Openstack driver.
- Do not alter users known\_hosts file in Openstack driver.
- Allow using environment variables in molecule.yml.
- Make ansible.cfg settings configurable through molecule.yml.
- Add multiple network support in Openstack driver.
- Add links functionality to Docker driver.
- Switched options from 'sudo' to 'become'.

### 8.1.55 1.17.3

- Create test skeleton with *molecule init* when initializing a role in current directory.

### 8.1.56 1.17.2

- Fix unittests to allow ls to be in both /usr/bin and /bin.
- Force raw\_env\_vars to string for *ansible-playbook*.

### 8.1.57 1.17.1

- Correct functional tests.
- Correct locale issues with print class of methods.
- Correct ansible-lint exit error when role dependency is in newer dictionary format.
- Pass env to *ansible-lint*.

### 8.1.58 1.17

- Cleanup sphinx doc generation.
- Bumped testinfra requirement which drops the now useless installation of which in centos and fedora images.
- Made OpenStack's ip pool configurable.
- Corrected Docker's overlayfs for RPM based distros.
- Fixed OpenStack's security\_groups default for newer shade versions.
- Added missing bash completion targets.

### 8.1.59 1.16.1

- Removed check mode from running in test cycle.



## Breaking Changes

- Molecule no longer runs in “Dry Mode” as part of *molecule test*. If one wishes to incorporate check as part of *test*, *molecule.yml* can be updated to include this as part of the test sequence.

### 8.1.60 1.16

- Slightly improved unit test coverage.
- Various doc improvements.
- Added Gilt usage to docs.
- Reimplemented info, error, debug message handling.
- Nice error message when rake and/or rubocop missing.
- Fix task determination on idempotence failure.
- Added a github issue template.
- Logging of dependency command execution.

### 8.1.61 1.15

- Added a shell dependency manager.
- Created a CI section to documentation with Tox details.
- Rename dependencies key to dependency.

## Breaking Changes

- The galaxy override options have been moved to the *dependency* section of molecule’s config. No longer support a top level *dependencies* config key. This functionality was added in 1.14, and this follow-up corrects the usage, before 1.14 was utilized.

### 8.1.62 1.14.1

- Fix openstack driver login and ssh key generation.

### 8.1.63 1.14

- Made improvements to unit/functional tests.
- Fixed Goss verifier under Ansible 2.2.
- Removed testinfra config backward compatibility.
- Broke out role dependency into a subcommand.

## Breaking Changes

- The `testinfra` override options have been moved to the `verifier` section of molecule's config. No longer support a top level `testinfra` config key.
- The `galaxy` override options have been moved to the `dependencies` section of molecule's config. No longer support a `galaxy` key inside the top level `ansible` section.

### 8.1.64 1.13

- Implement environment handling in docker driver.
- Added `vmware_workstation` provider to `vagrant`.
- Improved overall logging, including logging of `sh` commands when debug flag set.
- Avoid images with `<none>` tag.
- Support and test ansible 2.2 and 2.1.2.
- Allow nested `testinfra` test directory structure.
- Ability to pass arbitrary ansible cli flags to `converge`.
- Added IRC info to docs.
- Return exit code from goss verifier.
- General cleanup of modules and documentation.
- Bumped requirements versions.

### 8.1.65 1.12.6

- Disable diff when executing idempotent check.
- Make sure `ansible-lint` respects the molecule `ignore_paths`.
- Convert `readthedocs` links for their `.org->.io` migration for hosted projects.

### 8.1.66 1.12.5

- Increased test coverage.
- Allow `group/host` vars in `molecule.yml` to work with ansible 1.9.
- Pass `HOME` to `ansible-lint` environment.
- Expose driver to login.
- Improved login error message messaging.

### 8.1.67 1.12.4

- Added a private disabled top level key. Do not use or rely on this key. Added for our molecule adoption.
- Added a coverage minimum.
- More unit and functional coverage.

### 8.1.68 1.12.3

- Write templates even when a custom `ansible.cfg` is specified.

### 8.1.69 1.12.2

- Removed default multiple-instances from `init`.

### 8.1.70 1.12.1

- Preserve `ansible.cfg` when supplying a custom one.

### 8.1.71 1.12

- Additional command tests.
- Changed connection to `ansible_connection`.
- Implemented `click` vs `docopt`. This slightly changes the CLI's semantics.
- Removed the driver python packages from installing with molecule.
- Set ssh key if specified in OpenStack driver.
- Using `py.test` as functional test runner.
- Added a Gemfile to `molecule init serverspec verifier`.
- Added SUSE docker driver support.
- Display the list of non-idempotent tasks with `molecule idempotence`.

### Breaking Changes

- The `--debug` flag is no longer passed to the subcommand. The command and subcommand args were getting munged together, and passed to the core. They are now handled separately.
- Removed the `--debug` subcommand flag from all usage – it was never used.
- The `init` subcommand requires an optional `--role` flag vs a role argument when naming the role to initialize.
- The `init` subcommand requires a `--driver` flag when creating a driver other than `vagrant`.
- The `init` subcommand requires a `--verifier` flag when creating a verifier other than `testinfra`.
- The `login` subcommand requires a `--host` flag when more than one instance exists.
- One must install the appropriate python package based on the driver used.

### 8.1.72 1.11.5

- Set ssh key if specified with the OpenStack driver.
- Pass `ANSIBLE_CONFIG` when executing `ansible-lint`.

### 8.1.73 1.11.4

- Hide ansible-lint stacktrace on `molecule verify`.
- Corrected linked clone platform options checking.

### 8.1.74 1.11.3

- Handle when a container is stopped outside of molecule, when running `molecule status`.

### 8.1.75 1.11.2

- Preserve sudo passed in verifier options.

### 8.1.76 1.11.1

- Corrected bug when passing the `--platform` flag.

### 8.1.77 1.11

- General cleanup of core module.
- Various documentation updates.
- Pull molecule status from state file when using Vagrant driver.
- Added alpha Goss verifier support.
- Updated runtime requirements to current versions.
- Implemented `molecule check` subcommand.
- Configure verifier to be test kitchen like.
- Ability to declare multiple drivers in config.
- Implement ansible groups inheritance.

## Breaking Changes

Previously molecule would execute a test framework based on the existence of a directory structure. This is no longer the case. Molecule will execute the configured suite, where *testinfra* is the default. See docs.

### 8.1.78 1.10.3

- Reimplemented idempotence handling. Removed the idempotence ansible callback plugin, in favor of a native implementation.

## Note

There is no change in workflow. Molecule still reports if a converge was idempotent or not. However, it no longer reports which task(s) are not idempotent.

### 8.1.79 1.10.2

- Removed `pytest-xdist` from runtime deps. This allows `testinfra`'s dependency on `pytest` to properly install.

### 8.1.80 1.10.1

- Pinned to explicit version of `testinfra`, due to `pytest` incompatibilities.

### 8.1.81 1.10

- Added ability to specify custom `dockerfile`.
- Added ability to generate and destroy temporary `openstack` `keypair` and `ssh` key file if they are not specified in the `molecule.yml`.
- Implemented `Cookiecutter` for `molecule init`.
- Documentation improvements.

### Breaking Changes

Roles may fail to converge due to the introduction of additional verifiers.

- Added `flake8` linter to `testinfra` verifier.
- Implemented `ansible` lint.

### 8.1.82 1.9.1

- Correct a `converge --debug` bug.
- Correct `ansible` galaxy role path.

### 8.1.83 1.9

- Restructured and reorganized internal code, tests, and docs.
- Added functional scenario tests.
- Improved unit tests/coverage.
- Added auto `docker` api version recognition to prevent api mismatch errors.
- Added fallback status for `vagrant` driver.
- Control over `ansible` galaxy options.
- Display `molecule` status when not created.
- Added dependency installation state, and installation step for syntax check.
- Pinned runtime requirements.
- Update `login` to use state data.
- Ability to target `ansible` groups with `testinfra`.
- Ability to target `docker` hosts with `serverspec`.

- Added ../../ to rolepath to fix ansible 2.1.1 default role search.
- Added docker volume mounting.
- Add support for Docker port bindings.
- Implemented a new core config class.

## Breaking Changes

- Existing Testinfra tests which use the Docker driver need updating as described in [PR #398](#).

### 8.1.84 1.8.4

- Fixed role\_path with ansible 2.1.1.

### 8.1.85 1.8.3

- Fixed passing flags to molecule test.

### 8.1.86 1.8.2

- Fixed a bad reference to the molecule\_dir config variable.

### 8.1.87 1.8.1

- Fixed a bug where molecule would fail if .molecule/ didn't already exist.

### 8.1.88 1.8

- Added native support for OpenStack provider.
- Fixed a bug where testinfra\_dir config option wasn't being handled.
- Fixed a bug with molecule login where its host matching didn't work with overlapping names.

### 8.1.89 1.7

- It's now possible to define host\_vars and group\_vars in ansible section of molecule.yml.
- The --platform CLI option now supports all.
- Corrected issue with specifying serverspec args in molecule.yml.

### 8.1.90 1.6.3

- Updated config parsing so that testinfra.sudo and testinfra.debug can be set in molecule.yml.
- Demo role now pulls in correct serverspec config.

### 8.1.91 1.6.2

- Added `inventory-file` flag to `molecule check` to address Ansible 1.9.x specific issue.

### 8.1.92 1.6.1

- Fixed a bug preventing `molecule test` from working.
- Added a demo role for functional testing.

### 8.1.93 1.6

- Added `--offline` option to `molecule init`.
- `molecule status` now shows hosts by default.
- `molecule test` will now fail immediately when encountering an error.
- Switched to Python's logging module for displaying `STDOUT`, `STDERR`.
- Added support for `libvirt` provider.
- Added `molecule check` to check playbook syntax.
- `Testinfra` parameters can now be set as vars in `molecule.yml`.
- Running `testinfra` tests in parallel is no longer the default behaviour.

### 8.1.94 1.5.1

- Fixed issue with `testinfra` and `serverspec` attempting to share args.
- Added `--sudo` option for `testinfra`.
- Added tab completion support.
- Misc. Docker updates and fixes.

### 8.1.95 1.5

- Added support for Docker provisioner.
- Added support for `group_vars`.

### 8.1.96 1.4.2

- Made `"append_platform_to_hostname"` `False` by default.
- `Testinfra` tests now run in parallel.
- `init` now generates `testinfra` tests by default.
- `Testinfra` env vars (including `ssh`) are now consistent with what is passed to `ansible-playbook`.

### 8.1.97 1.4.1

- Fixed a bug where `testinfra_dir` wasn't being used.
- Changed `append_platform_to_hostname` to default to `False`.

### 8.1.98 1.4

- Updated `init` to install role dependencies from Ansible Galaxy.
- Now using `DocOpt` subcommands to dispatch commands internally.
- Updated `login` command to take no hostname (for single instances) and partial hostnames.
- Improved visibility when running (and not running) tests.
- Can now pass multiple instances of `-tags` for specifying more than one tag.
- Can now pass `-destroy` flag to `test` with various options suitable for use in CI.
- Numerous small bug fixes.

### 8.1.99 1.3

- Added very basic support for the `vagrant-triggers` plugin.

### 8.1.100 1.2.4

- Fixed a bug introduced in 1.2.3 preventing `init` from working.

### 8.1.101 1.2.3

- Fixed a bug where `destroy` would fail on VMs that hadn't been created. Caused errors running `test`.
- Moved `rubocop`, `rake`, and `testinfra` into validators. Added tests.
- Moved `ansible-playbook` logic out of core, commands and into a dedicated class. Added tests.
- Provisioner logic moved to its own class outside of core.

### 8.1.102 1.2.2

- Added a CLI option for the `list` command to make the output machine readable.
- Refactored `commands.py` to be more conducive to dispatch from `DocOpt` (#76).
- Fixed #82 where callback plugin path wasn't being properly merged with user-defined values.
- Fixed #84 where `molecule init` would produce a `molecule.yml` that contained trailing whitespace.
- Fixed #85 preventing user-defined `serverspec` directory from being used.



### 8.1.103 1.2.1

- Updated idempotence plugin path to be appended to existing plugin path rather than overwriting it.
- Fixed case where idempotence plugin would crash when unable to read response dictionary.

### 8.1.104 1.2

- Added support for Vagrant 1.8's `linked_clone` option.
- Updated idempotence test to use an Ansible callback plugin that will print failed tasks.
- Path to templates can now be relative to a user's home directory.
- `box_url` in `Vagrantfile` is no longer set if `box_version` is defined.
- Uses the latest version of `python-vagrant`.

### 8.1.105 1.1.3

- Fixed a bug where inventory wasn't getting created on a new converge.
- Linting now targets a specific list of file extensions.
- Hostname created during `init` is now sanitized.
- Creation of python cache directory is now disabled by default.

### 8.1.106 1.1.2

- Fixed a bug where calling `create` separately from `converge` wasn't generating an inventory file.

### 8.1.107 1.1.1

- Cleaned up state file management logic to be more concise, functional for other purposes.
- Removed `-fast` flag from `converge` in favor of using state file for fast converging.
- Instance hostname is now printed during `serverspec` runs.
- Fixed a bug where loading template files from absolute paths didn't work.

### 8.1.108 1.1

- Added support for static inventory where molecule can manage existing sites, not just vagrant instances.
- Added support for skipping instance/inventory creation during `molecule converge` by passing it `-fast`. MUCH faster.

### 8.1.109 1.0.6

- Fixed a bug preventing vagrant raw\_config\_args from being written to vagrantfile template.
- Cleaned up error messaging when attempting to *molecule login* to a non-existent host.
- Added release engineering documentation.
- Moved commands into a separate module.
- Switched to using `yaml.safe_load()`.
- Added more debugging output.

### 8.1.110 1.0.5

- Added support for Vagrant box versioning. This allows teams to ensure all members are using the correct version in their development environments.

### 8.1.111 1.0.4

- Fixed a bug where specifying an inventory script was causing molecule to create it.
- `config_file` and `inventory_file` specified in ansible block are now treated as overrides for molecule defaults.

### 8.1.112 1.0.3

- Updated format of `config.yml` and `molecule.yml` so they use the same data structure for easier merging. In general it's more clear and easy to understand.
- Defaults are now loaded from a defaults file (YAML) rather than a giant hash. Maintaining data in two formats was getting tiresome.
- Decoupled `main()` from `init()` in Molecule core to make future tests easier.
- Removed mock from existing tests that no longer require it now that `main()` is decoupled.
- Moved all config handling to an external class. Greatly simplified all logic.
- Added tests for new config class.
- Cleaned up all messages using `format()` to have consistent syntax.
- Fixed status command to not fire unless a vagrantfile is present since it was triggering vagrant errors.
- Renamed `_init_new_role()` to `init()` to be consistent with other commands.
- Fixed incorrect messaging in `_print_valid_providers()`.
- Fixed edge case in vagrantfile template to make sure we always have default cpus/memory set for virtualbox instances.
- Leveraged new config flexibility to clean up old hack for `molecule init`.
- Fixed utility test for `deep_merge` that was failing.
- Made `print_line` two different functions for `stdout` and `stderr`.
- Updated print functions to be Python 3 ready.
- Moved template creation into a generic function.

- Test all the (moved) things.
- Updated image assets.
- Removed aio/mcp naming from docs and templates.

### **8.1.113 1.0.2**

- Switched to deep merging of config dicts rather than using update().

### **8.1.114 1.0.1**

- Fixed trailing validator, and broke out into a module.

### **8.1.115 1.0**

- Initial release.



## Symbols

-molecule-init-scenario--scenario-name-bar  
 command line option; cd  
 foo; molecule init scenario  
 -scenario-name bar -role-name  
 foo  
 cd-foo, 15

-molecule-init-scenario--scenario-name-bar  
 command line option; cd  
 foo; molecule init scenario  
 -scenario-name bar -role-name  
 foo -driver-template path  
 cd-foo, 15

## A

Ansible (class in molecule.provisioner.ansible), 28  
 Ansible (class in molecule.verifier.ansible), 35  
 AnsibleGalaxy (class  
 molecule.dependency.ansible\_galaxy), 20  
 AnsibleLint (class  
 molecule.provisioner.lint.ansible\_lint), 33  
 AnsibleLint (class  
 molecule.verifier.lint.ansible\_lint), 36

## C

cd-foo  
 -molecule-init-scenario--scenario-name-bar  
 command line option; cd  
 foo; molecule init scenario  
 -scenario-name bar -role-name  
 foo, 15  
 -molecule-init-scenario--scenario-name-bar  
 command line option; cd  
 foo; molecule init scenario  
 -scenario-name bar -role-name  
 foo -driver-template path, 15  
 Check (class in molecule.command.check), 12  
 Cleanup (class in molecule.command.cleanup), 12  
 Config (class in molecule.config), 19  
 Converge (class in molecule.command.converge), 12  
 Create (class in molecule.command.create), 13

## D

Delegated (class in molecule.driver.delegated), 22  
 Dependency (class in molecule.command.dependency),  
 13  
 Destroy (class in molecule.command.destroy), 14  
 Docker (class in molecule.driver.docker), 23

## F

--role-name-foo--driver-template-path  
 Flake8 (class in molecule.verifier.lint.flake8), 37

## G

Gilt (class in molecule.dependency.gilt), 20

## I

Idempotence (class in  
 molecule.command.idempotence), 14  
 Interpolator (class in molecule.interpolation), 19

## L

in  
 Lint (class in molecule.command.lint), 15  
 List (class in molecule.command.list), 15  
 in  
 Login (class in molecule.command.login), 16

## M

Matrix (class in molecule.command.matrix), 16  
 molecule -base-config base.yml check  
 command line option, 12  
 molecule -base-config base.yml cleanup  
 molecule--base-config-base.yml-cleanup  
 command line option, 12  
 molecule -base-config base.yml  
 molecule --base-config base.yml  
 converge  
 molecule--base-config-base.yml-converge  
 command line option, 13  
 molecule -base-config base.yml create  
 molecule--base-config-base.yml-create  
 command line option, 13  
 molecule -base-config base.yml  
 dependency  
 molecule--base-config-base.yml-dependency  
 command line option, 13

molecule -base-config base.yml destroy molecule -debug idempotence  
 molecule--base-config-base.yml-destroy molecule--debug-idempotence  
 command line option, 14 command line option, 14

molecule -base-config base.yml idempotence molecule -debug lint  
 molecule--base-config-base.yml-idempotence option, 15 molecule--debug-lint command line  
 command line option, 14 molecule -debug list  
 molecule--debug-list command line  
 option, 16

molecule -base-config base.yml lint molecule -debug login  
 molecule--base-config-base.yml-lint command line option, 15 molecule--debug-login command line  
 option, 16

molecule -base-config base.yml list molecule -debug matrix subcommand  
 molecule--base-config-base.yml-list command line option, 16 molecule--debug-matrix-subcommand  
 command line option, 16

molecule -base-config base.yml login molecule -debug prepare  
 molecule--base-config-base.yml-login command line option, 16 molecule--debug-prepare command  
 line option, 17

molecule -base-config base.yml matrix molecule -debug side-effect  
 molecule--base-config-base.yml-matrix-subcommand molecule--debug-side-effect  
 command line option, 16 molecule -debug syntax  
 molecule--debug-syntax command  
 line option, 18

molecule -base-config base.yml prepare molecule -debug test  
 molecule--base-config-base.yml-prepare command line option, 17 molecule--debug-test command line  
 option, 18

molecule -base-config base.yml side-effect molecule -debug verify  
 molecule--base-config-base.yml-side-effect command line option, 17 molecule--debug-verify command  
 line option, 18

molecule -base-config base.yml syntax molecule -env-file foo.yml check  
 molecule--base-config-base.yml-syntax command line option, 18 molecule--env-file-foo.yml-check  
 command line option, 12

molecule -base-config base.yml test molecule -env-file foo.yml cleanup  
 molecule--base-config-base.yml-test command line option, 18 molecule--env-file-foo.yml-cleanup  
 command line option, 12

molecule -base-config base.yml verify molecule -env-file foo.yml converge  
 molecule--base-config-base.yml-verify command line option, 18 molecule--env-file-foo.yml-converge  
 command line option, 13

molecule -debug check molecule -env-file foo.yml create  
 molecule--debug-check command line option, 12 molecule--env-file-foo.yml-create  
 command line option, 13

molecule -debug cleanup molecule -env-file foo.yml dependency  
 molecule--debug-cleanup command line option, 12 molecule--env-file-foo.yml-dependency  
 command line option, 14

molecule -debug converge molecule -env-file foo.yml destroy  
 molecule--debug-converge command line option, 13 molecule--env-file-foo.yml-destroy  
 command line option, 14

molecule -debug create molecule -env-file foo.yml idempotence  
 molecule--debug-create command line option, 13 molecule--env-file-foo.yml-idempotence  
 command line option, 14

molecule -debug dependency molecule -env-file foo.yml lint  
 molecule--debug-dependency command line option, 13 molecule--env-file-foo.yml-lint  
 command line option, 15

molecule -debug destroy molecule--debug-destroy command  
 line option, 14

molecule -env-file foo.yml list  
   molecule--env-file-foo.yml-list  
     command line option, 16

molecule -env-file foo.yml login  
   molecule--env-file-foo.yml-login  
     command line option, 16

molecule -env-file foo.yml matrix  
   subcommand  
   molecule--env-file-foo.yml-matrix-subcommand  
     command line option, 16

molecule -env-file foo.yml prepare  
   molecule--env-file-foo.yml-prepare  
     command line option, 17

molecule -env-file foo.yml side-effect  
   molecule--env-file-foo.yml-side-effect  
     command line option, 17

molecule -env-file foo.yml syntax  
   molecule--env-file-foo.yml-syntax  
     command line option, 18

molecule -env-file foo.yml test  
   molecule--env-file-foo.yml-test  
     command line option, 18

molecule -env-file foo.yml verify  
   molecule--env-file-foo.yml-verify  
     command line option, 18

molecule -parallel check  
   molecule--parallel-check command  
   line option, 12

molecule -parallel destroy  
   molecule--parallel-destroy command  
   line option, 14

molecule -parallel test  
   molecule--parallel-test command  
   line option, 18

molecule check  
   molecule-check command line option,  
   12

molecule check -scenario-name foo  
   molecule-check--scenario-name-foo  
   command line option, 12

molecule cleanup  
   molecule-cleanup command line  
   option, 12

molecule cleanup -scenario-name foo  
   molecule-cleanup--scenario-name-foo  
   command line option, 12

molecule converge  
   molecule-converge command line  
   option, 13

molecule converge - -vvv -tags foo,bar  
   molecule-converge---vvv--tags-foo,bar  
   command line option, 13

molecule converge -scenario-name foo  
   molecule-converge--scenario-name-foo  
   command line option, 13

molecule create  
   molecule-create command line  
   option, 13

molecule create -driver-name foo  
   molecule-create--driver-name-foo  
   command line option, 13

molecule create -scenario-name foo  
   molecule-create--scenario-name-foo  
   command line option, 13

molecule dependency  
   molecule-dependency command line  
   option, 13

molecule dependency -scenario-name foo  
   molecule-dependency--scenario-name-foo  
   command line option, 13

molecule destroy  
   molecule-destroy command line  
   option, 14

molecule destroy -all  
   molecule-destroy--all command line  
   option, 14

molecule destroy -driver-name foo  
   molecule-destroy--driver-name-foo  
   command line option, 14

molecule destroy -scenario-name foo  
   molecule-destroy--scenario-name-foo  
   command line option, 14

molecule idempotence  
   molecule-idempotence command line  
   option, 14

molecule idempotence -scenario-name  
   foo  
   molecule-idempotence--scenario-name-foo  
   command line option, 14

molecule init role -role-name foo  
   molecule-init-role--role-name-foo  
   command line option, 15

molecule init role -role-name foo  
   -template path  
   molecule-init-role--role-name-foo--template-pat  
   command line option, 15

molecule init scenario -scenario-name  
   bar -role-name foo  
   molecule-init-scenario--scenario-name-bar--role  
   command line option, 15

molecule init template -url  
   https://example.com/user/cookiecutter-repo  
   molecule-init-template--url-https://example.com  
   command line option, 15

molecule lint  
   molecule-lint command line option,  
   15

molecule lint -scenario-name foo

```

    molecule-lint--scenario-name-foo
        command line option, 15
molecule list
    molecule-list command line option,
    15
molecule list -format plain
    molecule-list--format-plain
        command line option, 16
molecule list -format yaml
    molecule-list--format-yaml command
    line option, 16
molecule list -scenario-name foo
    molecule-list--scenario-name-foo
        command line option, 15
molecule login
    molecule-login command line option,
    16
molecule login -host hostname
    molecule-login--host-hostname
        command line option, 16
molecule login -host hostname
    -scenario-name foo
    molecule-login--host-hostname--scenario-name-foo
        command line option, 16
molecule login -scenario-name foo
    molecule-login--scenario-name-foo
        command line option, 16
molecule matrix -scenario-name foo
    subcommand
    molecule-matrix--scenario-name-foo-subcommand
        command line option, 16
molecule matrix subcommand
    molecule-matrix-subcommand command
    line option, 16
molecule prepare
    molecule-prepare command line
    option, 17
molecule prepare -driver-name foo
    molecule-prepare--driver-name-foo
        command line option, 17
molecule prepare -force
    molecule-prepare--force command
    line option, 17
molecule prepare -scenario-name foo
    molecule-prepare--scenario-name-foo
        command line option, 17
molecule side-effect
    molecule-side-effect command line
    option, 17
molecule side-effect -scenario-name
    foo
    molecule-side-effect--scenario-name-foo
        command line option, 17
molecule syntax
    molecule-syntax command line
    option, 17
molecule syntax -scenario-name foo
    molecule-syntax--scenario-name-foo
        command line option, 17
molecule test
    molecule-test command line option,
    18
molecule test -all
    molecule-test--all command line
    option, 18
molecule test -destroy=always
    molecule-test--destroy=always
        command line option, 18
molecule test -scenario-name foo
    molecule-test--scenario-name-foo
        command line option, 18
molecule verify
    molecule-verify command line
    option, 18
molecule verify -scenario-name foo
    molecule-verify--scenario-name-foo
        command line option, 18
molecule --base-config-base.yml-check
    command line option
    molecule -base-config base.yml
    check, 12
molecule --base-config-base.yml-cleanup
    command line option
    molecule -base-config base.yml
    cleanup, 12
molecule --base-config-base.yml-converge
    command line option
    molecule -base-config base.yml
    converge, 13
molecule --base-config-base.yml-create
    command line option
    molecule -base-config base.yml
    create, 13
molecule --base-config-base.yml-dependency
    command line option
    molecule -base-config base.yml
    dependency, 13
molecule --base-config-base.yml-destroy
    command line option
    molecule -base-config base.yml
    destroy, 14
molecule --base-config-base.yml-idempotence
    command line option
    molecule -base-config base.yml
    idempotence, 14
molecule --base-config-base.yml-lint
    command line option
    molecule -base-config base.yml

```



lint, 15  
 molecule--base-config-base.yml-list  
   command line option  
   molecule -base-config base.yml  
   list, 16  
 molecule--base-config-base.yml-login  
   command line option  
   molecule -base-config base.yml  
   login, 16  
 molecule--base-config-base.yml-matrix-subcommand  
   command line option  
   molecule -base-config base.yml  
   matrix subcommand, 16  
 molecule--base-config-base.yml-prepare  
   command line option  
   molecule -base-config base.yml  
   prepare, 17  
 molecule--base-config-base.yml-side-effect  
   command line option  
   molecule -base-config base.yml  
   side-effect, 17  
 molecule--base-config-base.yml-syntax  
   command line option  
   molecule -base-config base.yml  
   syntax, 18  
 molecule--base-config-base.yml-test  
   command line option  
   molecule -base-config base.yml  
   test, 18  
 molecule--base-config-base.yml-verify  
   command line option  
   molecule -base-config base.yml  
   verify, 18  
 molecule--debug-check command line  
   option  
   molecule -debug check, 12  
 molecule--debug-cleanup command line  
   option  
   molecule -debug cleanup, 12  
 molecule--debug-converge command line  
   option  
   molecule -debug converge, 13  
 molecule--debug-create command line  
   option  
   molecule -debug create, 13  
 molecule--debug-dependency command  
   line option  
   molecule -debug dependency, 13  
 molecule--debug-destroy command line  
   option  
   molecule -debug destroy, 14  
 molecule--debug-idempotence command  
   line option  
   molecule -debug idempotence, 14  
 molecule--debug-lint command line  
   option  
   molecule -debug lint, 15  
 molecule--debug-list command line  
   option  
   molecule -debug list, 16  
 molecule--debug-login command line  
   option  
   molecule -debug login, 16  
 molecule--debug-matrix-subcommand  
   command line option  
   molecule -debug matrix subcommand,  
   16  
 molecule--debug-prepare command line  
   option  
   molecule -debug prepare, 17  
 molecule--debug-side-effect command  
   line option  
   molecule -debug side-effect, 17  
 molecule--debug-syntax command line  
   option  
   molecule -debug syntax, 18  
 molecule--debug-test command line  
   option  
   molecule -debug test, 18  
 molecule--debug-verify command line  
   option  
   molecule -debug verify, 18  
 molecule--env-file-foo.yml-check  
   command line option  
   molecule -env-file foo.yml check, 12  
 molecule--env-file-foo.yml-cleanup  
   command line option  
   molecule -env-file foo.yml cleanup,  
   12  
 molecule--env-file-foo.yml-converge  
   command line option  
   molecule -env-file foo.yml  
   converge, 13  
 molecule--env-file-foo.yml-create  
   command line option  
   molecule -env-file foo.yml create,  
   13  
 molecule--env-file-foo.yml-dependency  
   command line option  
   molecule -env-file foo.yml  
   dependency, 14  
 molecule--env-file-foo.yml-destroy  
   command line option  
   molecule -env-file foo.yml destroy,  
   14  
 molecule--env-file-foo.yml-idempotence  
   command line option  
   molecule -env-file foo.yml

idempotence, 14  
 molecule--env-file-foo.yml-lint  
     command line option  
     molecule -env-file foo.yml lint, 15  
 molecule--env-file-foo.yml-list  
     command line option  
     molecule -env-file foo.yml list, 16  
 molecule--env-file-foo.yml-login  
     command line option  
     molecule -env-file foo.yml login, 16  
 molecule--env-file-foo.yml-matrix-subcommand  
     command line option  
     molecule -env-file foo.yml matrix  
     subcommand, 16  
 molecule--env-file-foo.yml-prepare  
     command line option  
     molecule -env-file foo.yml prepare,  
     17  
 molecule--env-file-foo.yml-side-effect  
     command line option  
     molecule -env-file foo.yml  
     side-effect, 17  
 molecule--env-file-foo.yml-syntax  
     command line option  
     molecule -env-file foo.yml syntax,  
     18  
 molecule--env-file-foo.yml-test  
     command line option  
     molecule -env-file foo.yml test, 18  
 molecule--env-file-foo.yml-verify  
     command line option  
     molecule -env-file foo.yml verify,  
     18  
 molecule--parallel-check command line  
     option  
     molecule -parallel check, 12  
 molecule--parallel-destroy command  
     line option  
     molecule -parallel destroy, 14  
 molecule--parallel-test command line  
     option  
     molecule -parallel test, 18  
 molecule-check command line option  
     molecule check, 12  
 molecule-check--scenario-name-foo  
     command line option  
     molecule check -scenario-name foo,  
     12  
 molecule-cleanup command line option  
     molecule cleanup, 12  
 molecule-cleanup--scenario-name-foo  
     command line option  
     molecule cleanup -scenario-name  
     foo, 12  
 molecule-converge command line option  
     molecule converge, 13  
 molecule-converge---vvv--tags-foo,bar  
     command line option  
     molecule converge - -vvv -tags  
     foo,bar, 13  
 molecule-converge--scenario-name-foo  
     command line option  
     molecule converge -scenario-name  
     foo, 13  
 molecule-create command line option  
     molecule create, 13  
 molecule-create--driver-name-foo  
     command line option  
     molecule create -driver-name foo, 13  
 molecule-create--scenario-name-foo  
     command line option  
     molecule create -scenario-name foo,  
     13  
 molecule-dependency command line  
     option  
     molecule dependency, 13  
 molecule-dependency--scenario-name-foo  
     command line option  
     molecule dependency -scenario-name  
     foo, 13  
 molecule-destroy command line option  
     molecule destroy, 14  
 molecule-destroy--all command line  
     option  
     molecule destroy -all, 14  
 molecule-destroy--driver-name-foo  
     command line option  
     molecule destroy -driver-name foo,  
     14  
 molecule-destroy--scenario-name-foo  
     command line option  
     molecule destroy -scenario-name  
     foo, 14  
 molecule-idempotence command line  
     option  
     molecule idempotence, 14  
 molecule-idempotence--scenario-name-foo  
     command line option  
     molecule idempotence  
     -scenario-name foo, 14  
 molecule-init-role--role-name-foo  
     command line option  
     molecule init role -role-name foo,  
     15  
 molecule-init-role--role-name-foo--template-path  
     command line option  
     molecule init role -role-name foo  
     -template path, 15

molecule-init-scenario--scenario-name-bar--molecule-prepare -force, 17  
     command line option                      molecule-prepare--scenario-name-foo  
     molecule init scenario                      command line option  
     -scenario-name bar -role-name              molecule prepare -scenario-name  
     foo, 15                                      foo, 17

molecule-init-template--url-https://example.com/user/cookiecutnolecule-side-effect, 17  
     command line option                      option  
     molecule init template -url              molecule side-effect, 17  
     https://example.com/user/cookiecutnolecule-side-effect--scenario-name-foo  
     15    command line option  
   molecule side-effect  
   -scenario-name foo, 17

molecule-lint command line option              molecule-syntax command line option  
     molecule lint, 15                          molecule syntax, 17

molecule-lint--scenario-name-foo              molecule-syntax--scenario-name-foo  
     command line option                      command line option  
     molecule lint -scenario-name foo, 15      molecule syntax -scenario-name foo,  
   17

molecule-list command line option              molecule-test command line option  
     molecule list, 15                          molecule test, 18

molecule-list--format-plain command           molecule-test--all command line option  
     line option                                  molecule test -all, 18

    molecule list -format plain, 16  
 molecule-list--format-yaml command           molecule-test--destroy=always command  
     line option                                  line option  
     molecule list -format yaml, 16  
 molecule-list--scenario-name-foo              molecule-test--scenario-name-foo  
     command line option                      command line option  
     molecule list -scenario-name foo, 15      molecule test -scenario-name foo, 18

molecule-login command line option            molecule-verify command line option  
     molecule login, 16                        molecule verify, 18

molecule-login--host-hostname command        molecule-verify--scenario-name-foo  
     line option                                  command line option  
     molecule login -host hostname, 16  
   molecule verify -scenario-name foo,  
   18

molecule-login--host-hostname--scenario-name-foo command line option  
     command line option                      molecule verify --scenario-name-foo  
     molecule login -host hostname            command line option  
     -scenario-name foo, 16                      molecule verify -scenario-name foo,  
   18

molecule-login--scenario-name-foo              command line option  
     command line option                      molecule verify --scenario-name-foo  
     molecule login -scenario-name foo,        command line option  
     16    molecule verify -scenario-name foo,  
   18

molecule-matrix--scenario-name-foo-subcommand 38  
     command line option                      Prepare (*class in molecule.command.prepare*), 17

    molecule matrix -scenario-name foo  
     subcommand, 16

molecule-matrix-subcommand command            Role (*class in molecule.command.init.role*), 15  
     line option                                  R

    molecule matrix subcommand, 16            S

molecule-prepare command line option           Scenario (*class in molecule.command.init.scenario*),  
     molecule prepare, 17                      15

molecule-prepare--driver-name-foo              Scenario (*class in molecule.scenario*), 34  
     command line option                      Shell (*class in molecule.dependency.shell*), 21

    molecule prepare -driver-name foo,        SideEffect (*class in molecule.command.side\_effect*),  
     17    17

molecule-prepare--force command line           State (*class in molecule.state*), 35  
     option                                      Syntax (*class in molecule.command.syntax*), 17

## T

Template (*class in molecule.command.init.template*),  
15

Test (*class in molecule.command.test*), 18

Testinfra (*class in molecule.verifier.testinfra*), 36

## V

Verify (*class in molecule.command.verify*), 18

## Y

Yamllint (*class in molecule.lint.yamllint*), 27