orphan:

Testing Ansible

This document describes how to:

- Run tests locally using `ansible-test`
- Extend

Requirements

There are no special requirements for running `ansible-test` on Python 2.7 or later. The `argparse` package is required for Python 2.6. The requirements for each `ansible-test` command are covered later.

Test Environments

Most `ansible-test` commands support running in one or more isolated test environments to simplify testing.

Remote

The `--remote` option runs tests in a cloud hosted environment. An API key is required to use this feature.

    Recommended for integration tests.

See the list of supported platforms and versions for additional details.

Environment Variables

When using environment variables to manipulate tests there some limitations to keep in mind. Environment variables are:

- Not propagated from the host to the test environment when using the `--docker` or `--remote` options.
- Not exposed to the test environment unless enabled in `test/lib/ansible_test/_internal/util.py` in the `common_environment` function.

    Example: `ANSIBLE_KEEP_REMOTE_FILES=1` can be set when running `ansible-test integration --venv`. However, using the `--docker` option would require running `ansible-test shell` to gain access to the Docker environment. Once at the shell prompt, the environment variable could be set and the tests executed. This is useful for debugging tests inside a container by following the :ref:`Debugging AnsibleModule-based modules <debugging_modules>` instructions.

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]testing_running_locally.rst, line 49); *backlink*

    Unknown interpreted text role "ref".

Interactive Shell

Use the `ansible-test shell` command to get an interactive shell in the same environment used to run tests. Examples:

- `ansible-test shell --docker` - Open a shell in the default docker container.
- `ansible-test shell --venv --python 3.6` - Open a shell in a Python 3.6 virtual environment.

Code Coverage

Code coverage reports make it easy to identify untested code for which more tests should be written. Online reports are available but only cover the `devel` branch (see :ref:`developing_testing`). For new code local reports are needed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]testing_running_locally.rst, line 66); *backlink*

Unknown interpreted text role "ref".

Add the `--coverage` option to any test command to collect code coverage data. If you aren't using the `--venv` or `--docker` options which create an isolated python environment then you may have to use the `--requirements` option to ensure that the correct version of the coverage module is installed:

```
ansible-test coverage erase
ansible-test units --coverage apt
ansible-test integration --coverage aws_lambda
ansible-test coverage html
```

Reports can be generated in several different formats:

- `ansible-test coverage report` - Console report.
- `ansible-test coverage html` - HTML report.
- `ansible-test coverage xml` - XML report.

To clear data between test runs, use the `ansible-test coverage erase` command. For a full list of features see the online help:

```
ansible-test coverage --help
```