integration-aliases

Integration tests are executed by `ansible-test` and reside in directories under `test/integration/targets/`. Each test MUST have an `aliases` file to control test execution.

Aliases are explained in the following sections. Each alias must be on a separate line in an `aliases` file.

Groups

Tests must be configured to run in exactly one group. This is done by adding the appropriate group to the `aliases` file.

The following are examples of some of the available groups:

- `shippable/posix/group1`
- `shippable/windows/group2`
- `shippable/azure/group3`
- `shippable/aws/group1`
- `shippable/cloud/group1`

Groups are used to balance tests across multiple CI jobs to minimize test run time. They also improve efficiency by keeping tests with similar requirements running together.

When selecting a group for a new test, use the same group as existing tests similar to the one being added. If more than one group is available, select one randomly.

Setup

Aliases can be used to execute setup targets before running tests:

- `setup/once/TARGET` - Run the target `TARGET` before the first target that requires it.
- `setup/always/TARGET` - Run the target `TARGET` before each target that requires it.

Requirements

Aliases can be used to express some test requirements:

- `needs/privileged` - Requires `--docker-privileged` when running tests with `--docker`.
- `needs/root` - Requires running tests as `root` or with `--docker`.
- `needs/ssh` - Requires SSH connections to localhost (or the test container with `--docker`) without a password.
- `needs/httptester` - Requires use of the http-test-container to run tests.

Dependencies

Some test dependencies are automatically discovered:

- Ansible role dependencies defined in `meta/main.yml` files.
- Setup targets defined with `setup/*` aliases.
- Symbolic links from one target to a file in another target.

Aliases can be used to declare dependencies that are not handled automatically:

- `needs/target/TARGET` - Requires use of the test target `TARGET`.
- `needs/file/PATH` - Requires use of the file `PATH` relative to the git root.

Skipping

Aliases can be used to skip platforms using one of the following:

- `skip/freebsd` - Skip tests on FreeBSD.
- `skip/osx` - Skip tests on macOS.
- `skip/rhel` - Skip tests on RHEL.
- `skip/docker` - Skip tests when running in a Docker container.

Platform versions, as specified using the `--remote` option with `/` removed, can also be skipped:

- `skip/freebsd11.1` - Skip tests on FreeBSD 11.1.
- `skip/rhel7.6` - Skip tests on RHEL 7.6.

Windows versions, as specified using the `--windows` option can also be skipped:

- `skip/windows/2012` - Skip tests on Windows Server 2012.
- `skip/windows/2012-R2` - Skip tests on Windows Server 2012 R2.

Aliases can be used to skip Python major versions using one of the following:

- `skip/python2` - Skip tests on Python 2.x.
- `skip/python3` - Skip tests on Python 3.x.

For more fine grained skipping, use conditionals in integration test playbooks, such as:

```
when: ansible_distribution in ('Ubuntu')
```

Miscellaneous

There are several other aliases available as well:

- `destructive` - Requires `--allow-destructive` to run without `--docker` or `--remote`.
- `hidden` - Target is ignored. Usable as a dependency. Automatic for `setup_` and `prepare_` prefixed targets.

Unstable

Tests which fail sometimes should be marked with the `unstable` alias until the instability has been fixed. These tests will continue to run for pull requests which modify the test or the module under test.

This avoids unnecessary test failures for other pull requests, as well as tests on merge runs and nightly CI jobs.

There are two ways to run unstable tests manually:

- Use the `--allow-unstable` option for `ansible-test`
- Prefix the test name with `unstable/` when passing it to `ansible-test`.

Tests will be marked as unstable by a member of the Ansible Core Team. GitHub issues will be created to track each unstable test.

Disabled

Tests which always fail should be marked with the `disabled` alias until they can be fixed.

Disabled tests are automatically skipped.

There are two ways to run disabled tests manually:

- Use the `--allow-disabled` option for `ansible-test`
- Prefix the test name with `disabled/` when passing it to `ansible-test`.

Tests will be marked as disabled by a member of the Ansible Core Team. GitHub issues will be created to track each disabled test.

Unsupported

Tests which cannot be run in CI should be marked with the `unsupported` alias. Most tests can be supported through the use of simulators and/or cloud plugins.

However, if that is not possible then marking a test as unsupported will prevent it from running in CI.

There are two ways to run unsupported tests manually:

- Use the `--allow-unsupported` option for `ansible-test`
- Prefix the test name with `unsupported/` when passing it to `ansible-test`.

Tests will be marked as unsupported by the contributor of the test.

Cloud

Tests for cloud services and other modules that require access to external APIs usually require special support for testing in CI.

These require an additional alias to indicate the required test plugin.

Some of the available aliases are:

- `cloud/aws`
- `cloud/azure`
- `cloud/cs`
- `cloud/digitalocean`
- `cloud/foreman`
- `cloud/openshift`
- `cloud/tower`
- `cloud/vcenter`

Untested

Every module and plugin should have integration tests, even if the tests cannot be run in CI.

Issues

Tests that are marked as unstable or disabled will have an issue created to track the status of the test. Each issue will be assigned to one of the following projects:

- AWS
- Azure
- Windows
- General

Questions

For questions about integration tests reach out to @mattclay or @gundalow on GitHub or the `#ansible-devel` chat channel (using Matrix at ansible.im or using IRC at irc.libera.chat).