Creating new integration tests

This section covers the following cases:

- There are no integration tests for a collection / group of modules in a collection at all.
- You are adding a new module and you want to include integration tests.
- You want to add integration tests for a module that already exists without integration tests.

In other words, there are currently no tests for a module regardless of whether the module exists or not.

If the module already has tests, see :ref:`collection_updating_integration_tests`.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_integration_add.rst, **line 14**); *backlink*
>
> Unknown interpreted text role "ref".

Simplified example

Here is a simplified abstract example.

Let's say we are going to add integration tests to a new module in the `community.abstract` collection which interacts with some service.

We :ref:`checked<collection_integration_prepare>` and determined that there are no integration tests at all.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_integration_add.rst, **line 23**); *backlink*
>
> Unknown interpreted text role "ref".

We should basically do the following:

1. Install and run the service with a `setup` target.
2. Create a test target.
3. Add integration tests for the module.
4. :ref:`Run the tests<collection_run_integration_tests>`.

   > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_integration_add.rst, **line 30**); *backlink*
   >
   > Unknown interpreted text role "ref".

5. Fix the code and tests as needed, run the tests again, and repeat the cycle until they pass.

> **Note**
>
> You can reuse the `setup` target when implementing other targets that also use the same service.

1. Clone the collection to the `~/ansible_collections/community.abstract` directory on your local machine.
2. From the `~/ansble_collections/community.abstract` directory, create directories for the `setup` target:

```
mkdir -p tests/integration/targets/setup_abstract_service/tasks
```

3. Write all the tasks needed to prepare the environment, install, and run the service.

For simplicity, let's imagine that the service is available in the native distribution repositories and no sophisticated environment configuration is required.

Add the following tasks to the `tests/integration/targets/setup_abstract_service/tasks/main.yml` file to install and run the service:

```
- name: Install abstract service
  package:
    name: abstract_service

- name: Run the service
  systemd:
    name: abstract_service
    state: started
```

This is a very simplified example.

4. Add the target for the module you are testing.

Let's say the module is called `abstact_service_info`. Create the following directory structure in the target:

```
mkdir -p tests/integration/targets/abstract_service_info/tasks
mkdir -p tests/integration/targets/abstract_service_info/meta
```

Add all of the needed subdirectories. For example, if you are going to use defaults and files, add the `defaults` and `files` directories, and so on. The approach is the same as when you are creating a role.

5. To make the `setup_abstract_service` target run before the module's target, add the following lines to the `tests/integration/targets/abstract_service_info/meta/main.yml` file.

```
dependencies:
  - setup_abstract_service
```

6. Start with writing a single stand-alone task to check that your module can interact with the service.

We assume that the `abstract_service_info` module fetches some information from the `abstract_service` and that it has two connection parameters.

Among other fields, it returns a field called `version` containing a service version.

Add the following to `tests/integration/targets/abstract_service_info/tasks/main.yml`:

```
- name: Fetch info from abstract service
  anstract service info:
    host: 127.0.0.1  # We assume the service accepts local connection by default
    port: 1234       # We assume that the service is listening this port by default
  register: result   # This variable will contain the returned JSON including the server version

- name: Test the output
  assert:
    that:
      - result.version == '1.0.0'  # Check version field contains what we expect
```

7. :ref:`Run the tests<collection_run_integration_tests>` with the `-vvv` argument.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_integration_add.rst`, line 103); *backlink*
>
> Unknown interpreted text role "ref".

If there are any issues with connectivity (for example, the service is not accepting connections) or with the code, the play will fail.

Examine the output to see at which step the failure occurred. Investigate the reason, fix, and run again. Repeat the cycle until the test passes.

8. If the test succeeds, write more tests. Refer to the :ref:`Recommendations on coverage<collection_integration_recommendations>` section for details.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_integration_add.rst`, line 109); *backlink*
>
> Unknown interpreted text role "ref".

`community.postregesql` example

Here is a real example of writing integration tests from scratch for the `community.postgresql.postgresql_info` module.

For the sake of simplicity, we will create very basic tests which we will run using the Ubuntu 20.04 test container.

We use `Linux` as a work environment and have `git` and `docker` installed and running.

We also installed `ansible-core`.

1. Create the following directories in your home directory:

```
mkdir -p ~/ansible_collections/community
```

2. Fork the collection repository through the GitHub web interface.
3. Clone the forked repository from your profile to the created path:

```
git clone https://github.com/YOURACC/community.postgresql.git ~/ansible_collections/community/postgresql
```

If you prefer to use the SSH protocol:

```
git clone git@github.com:YOURACC/community.postgresql.git ~/ansible_collections/community/postgresql
```

4. Go to the cloned repository:

```
cd ~/ansible_collections/community/postgresql
```

5. Be sure you are in the default branch:

```
git status
```

6. Checkout a test branch:

```
git checkout -b postgresql_info_tests
```

7. Since we already have tests for the `postgresql_info` module, we will run the following command:

```
rm -rf tests/integration/targets/*
```

With all of the targets now removed, the current state is as if we do not have any integration tests for the `community.postgresql` collection at all. We can now start writing integration tests from scratch.

8. We will start with creating a `setup` target that will install all required packages and will launch PostgreSQL. Create the following directories:

```
mkdir -p tests/integration/targets/setup_postgresql_db/tasks
```

9. Create the `tests/integration/targets/setup_postgresql_db/tasks/main.yml` file and add the following tasks to it:

```
- name: Install required packages
  package:
    name:
      - apt-utils
      - postgresql
      - postgresql-common
      - python3-psycopg2

- name: Initialize PostgreSQL
  shell: . /usr/share/postgresql-common/maintscripts-functions && set_system_locale && /usr/bin/pg_createcluster -u p
  args:
    creates: /etc/postgresql/12/

- name: Start PostgreSQL service
  service:
    name: postgresql
    state: started
```

That is enough for our very basic example.

10. Then, create the following directories for the `postgresql_info` target:

```
mkdir -p tests/integration/targets/postgresql_info/tasks tests/integration/targets/postgresql_info/meta
```

11. To make the `setup_postgresql_db` target running before the `postgresql_info` target as a dependency, create the `tests/integration/targets/postgresql_info/meta/main.yml` file and add the following code to it:

```
dependencies:
  - setup_postgresql_db
```

12. Now we are ready to add our first test task for the `postgresql_info` module. Create the `tests/integration/targets/postgresql_info/tasks/main.yml` file and add the following code to it:

```
- name: Test postgresql_info module
  become: yes
  become_user: postgres
  postgresql_info:
    login user: postgres
    login_db: postgres
  register: result

- name: Check the module returns what we expect
  assert:
    that:
      - result is not changed
      - result.version.major == 12
      - result.version.minor == 8
```

In the first task, we run the `postgresql_info` module to fetch information from the database we installed and launched with the `setup_postgresql_db` target. We are saving the values returned by the module into the `result` variable.

In the second task, we check the `result` variable, which is what the first task returned, with the `assert` module. We expect that, among other things, the result has the version and reports that the system state has not been changed.

13. Run the tests in the Ubuntu 20.04 docker container:

```
ansible-test integration postgresql_info --docker ubuntu2004 -vvv
```

The tests should pass. If we look at the output, we should see something like the following:

```
TASK [postgresql_info : Check the module returns what we expect] ***************
ok: [testhost] => {
  "changed": false,
  "msg": "All assertions passed"
}
```

If your tests fail when you are working on your project, examine the output to see at which step the failure occurred. Investigate the reason, fix, and run again. Repeat the cycle until the test passes. If the test succeeds, write more tests. Refer to the :ref:`Recommendations on coverage<collection_integration_recommendations>` section for details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_integration_add.rst`, **line 250**); *backlink*

Unknown interpreted text role "ref".