Ansible Integration Function for IBM Resilient

Table of Contents

About This Package:

This package contains functions that integrate with Ansible to run playbooks and modules for remote host execution.

History

1.2 - Support for App Host 1.1 - change result payload format to JSON returned to Resilient

- For customers upgrading from 1.0, the example workflows will use different post-processing script logic, based on the json results returned.

Features:

- Run pre-written Ansible Playbooks from the Resilient platform, using parameter substitution.
- Run Ansible Modules for ad-hoc command execution.
- Run Resilient Rules at the Incident and Artifact levels.

Additional documentation on Ansible can be found at the Ansible website.

Package Components

- Functions:
  - fn_ansible
  - fn_ansible_module
- Workflows:
  - Example: Ansible - Run a Module
  - Example: Ansible - Run a Playbook from an Incident
  - Example: Ansible - Run a Playbook on an Artifact
- Rules:
  - Example: Ansible - Run a Module
  - Example: Ansible - Run a Playbook
  - Example: Ansible - Run a Playbook from an Artifact

App Host Installation

All the components for running Ansible in a container already exist when using the App Host container. The remainder of this section details the Ansible configuration file changes.

Under the Configuration Tab for an App, build out the ansible-runner files needed per the ansible-runner convention. The necessary files are the hosts file, ssh_key and your yml playbooks. Build these files under /var/rescircuits/ansible and ensure that the app.config file contains the same reference:

```
[fn_ansible]
runner_dir=/var/rescircuits/ansible
artifact_dir=/tmp
```

Playbooks

When adding playbooks through the Configuration tab of an App, ensure the File Path is /var/rescircuits/ansible/project.

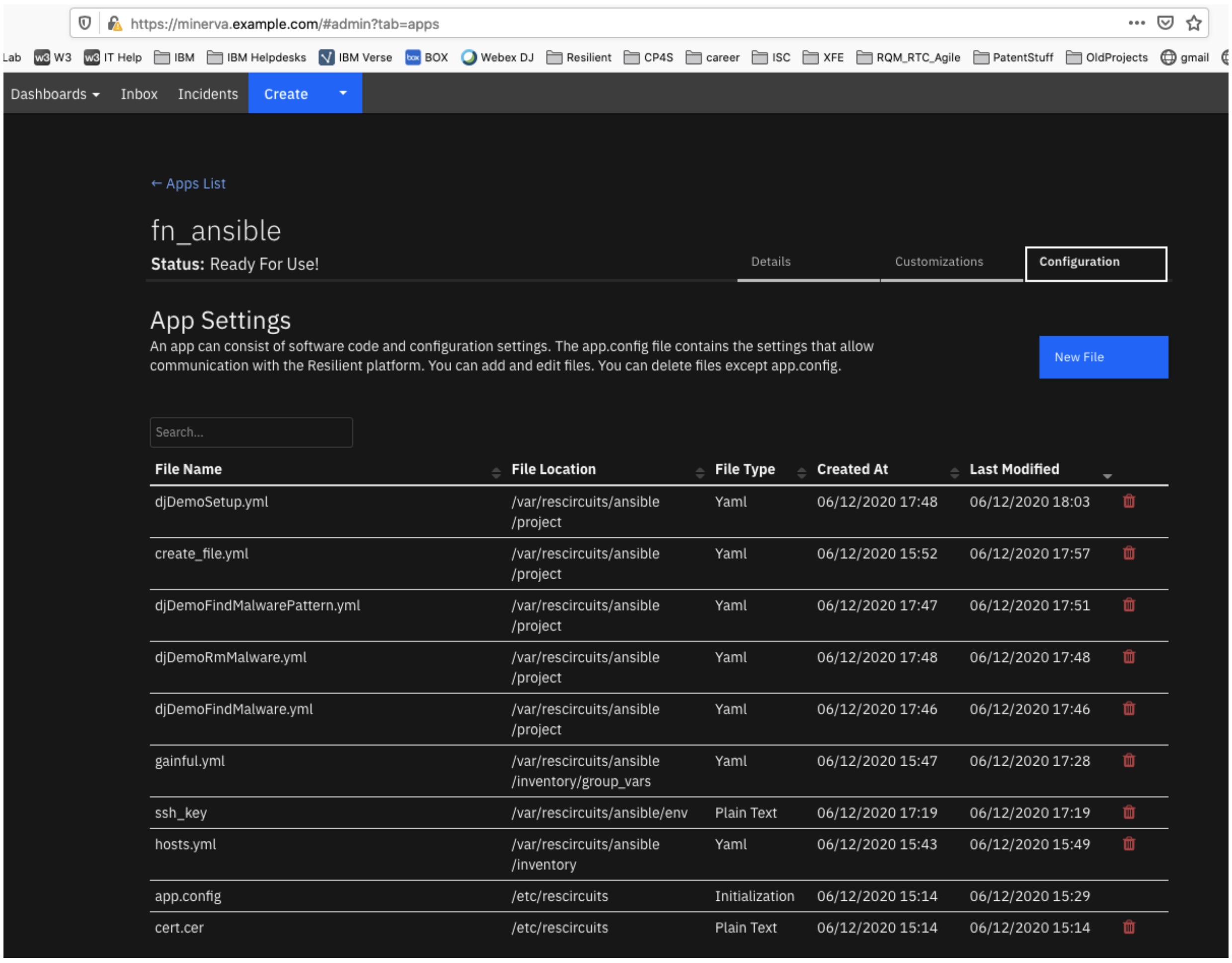
Note: Playbooks and modules cannot be run within the container environment.

The minimum Ansible files needed are:

- hosts, and
- playbooks

Your environment may require more configuration files, such as ssh_key and envvars.

This is an example of the configuration files used including several yaml files.


https://minerva.example.com/#admin?tab=apps
Lab W3 IT Help IBM IBM Helpdesks IBM Verse BOX Webex DJ Resilient CP4S career ISC XFE RQM_RTC_Agile PatentStuff OldProjects gmail
Dashboards
Inbox
Incidents
Create
← Apps List
Details
Customizations
Configuration
New File
Search...

fn_ansible

Status: Ready For Use!

App Settings

An app can consist of software code and configuration settings. The app.config file contains the settings that allow communication with the Resilient platform. You can add and edit files. You can delete files except app.config.

File Name	File Location	File Type	Created At	Last Modified
djDemoSetup.yml	/var/rescircuits/ansible /project	Yaml	06/12/2020 17:48	06/12/2020 18:03
create_file.yml	/var/rescircuits/ansible /project	Yaml	06/12/2020 15:52	06/12/2020 17:57
djDemoFindMalwarePattern.yml	/var/rescircuits/ansible /project	Yaml	06/12/2020 17:47	06/12/2020 17:51
djDemoRmMalware.yml	/var/rescircuits/ansible /project	Yaml	06/12/2020 17:48	06/12/2020 17:48
djDemoFindMalware.yml	/var/rescircuits/ansible /project	Yaml	06/12/2020 17:46	06/12/2020 17:46
gainful.yml	/var/rescircuits/ansible /inventory/group_vars	Yaml	06/12/2020 15:47	06/12/2020 17:28
ssh_key	/var/rescircuits/ansible/env	Plain Text	06/12/2020 17:19	06/12/2020 17:19
hosts.yml	/var/rescircuits/ansible /inventory	Yaml	06/12/2020 15:43	06/12/2020 15:49
app.config	/etc/rescircuits	Initialization	06/12/2020 15:14	06/12/2020 15:29
cert.cer	/etc/rescircuits	Plain Text	06/12/2020 15:14	06/12/2020 15:14

If you require additional ansible modules, additional effort is needed to include them as files in the Configuration tab.

Limitations

Presently, there are limitations in the use of containers when playbook parameters sent from Resilient are used with any file defined in /var/rescircuits/ansible/env.

Integration Server Installation

Prerequisites:

This integration relies on the installation of the ansible solution on the integration server. The process of installing ansible can be followed here.

- ansible >= 2.8.1
- ansible-runner >= 1.3.4
- Resilient platform >= v31.0.0
- Integrations Server Resilient Circuits >= v30.0.0
- Ansible config directory per the ansible-runner convention
- Ansible relies on a system library sshpass. Depending on your Integration Server operation system, different procedures are required to install this system library.
- For RHEL servers, the Red Hat Developer Toolset is needed to build the ansible runtime environment. This may also require a registered subscription manager to install.

This package requires that it is installed on a RHEL or CentOS platform and uses the Resilient Circuits framework.

- Unzip the package downloaded from the IBM App Exchange

```
$ unzip app-fn_ansible-<version>.zip
```

- To install the package, run:

  ```
  $ [sudo] pip install fn_ansible-<version>.tar.gz
  ```

- To import the function, example rules and workflows into your Resilient platform, run:

  ```
  $ resilient-circuits customize -l fn-ansible
  ```

- To update your app.config file with the required Ansible configurations, run:

  ```
  $ resilient-circuits config -u -l fn-ansible
  ```

- Within the app.config file, edit the following configuration data:

```
[fn_ansible]
runner_dir=</full/path/to/your/ansible/directory>
# temporary files collected when running a module or a playbook
artifact_dir=</full/path/to/artifacts/directory>
# change this value to trim the collection of previous process runs
artifact_retention_num=0
```

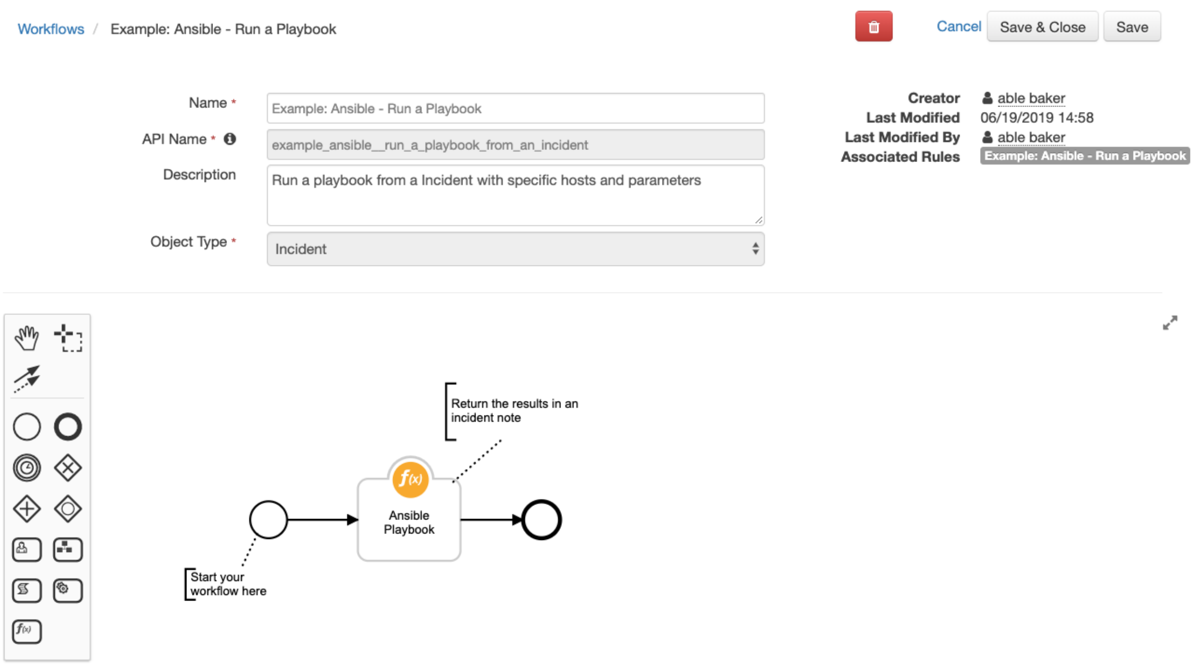
- To uninstall:

```
$ [sudo] pip uninstall fn-ansible
```

- Run Resilient Circuits:

```
$ resilient-circuits run
```

Sample Workflow:



Workflows / Example: Ansible - Run a Playbook

Name *	Example: Ansible - Run a Playbook
API Name *	example_ansible__run_a_playbook_from_an_incident
Description	Run a playbook from a Incident with specific hosts and parameters
Object Type *	Incident

Creator	able baker
Last Modified	06/19/2019 14:58
Last Modified By	able baker
Associated Rules	Example: Ansible - Run a Playbook

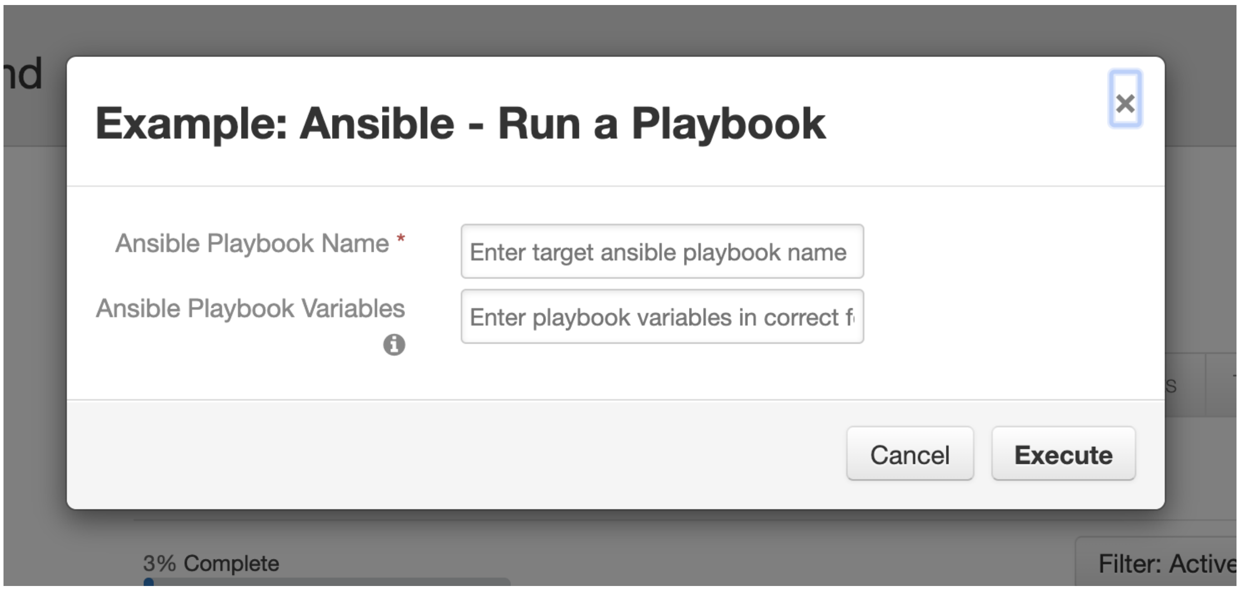
Function Inputs:

Input Name	Type	Required	Example	Info
ansible_playbook_name	String	Yes	"my_playbook" or "my_playbook.yml"	Identifies target playbook to run

Input Name	Type	Required	Example	Info
ansible_parameters	String	No	"host_names=192.168.1.4"	Substitution variables for a Playbook at runtime. For example, a Playbook containing hosts: "" will be replaced using the example value of 196.168.1.4. Use semi-colons (😉 to separate different parameters.

Some parameters can contain more than one value. In these cases, use a comma (,) to separate the values. Example: host_names=192.168.56.1,192.168.56.2.

When using playbooks run against artifacts, an additional parameter is sent specifying the artifact value to the playbook as artifact_value=<artifact_value>. Therefore, in your playbook, ensure to add for value substitution.


nd
Example: Ansible - Run a Playbook
Ansible Playbook Name *
Enter target ansible playbook name
Ansible Playbook Variables
Enter playbook variables in correct f
Cancel
Execute
3% Complete
Filter: Active

Function Output:

Results returned to the Resilient platform from an Ansible function are in the following payload pattern:

```
{
  'version': '1.0',
  'success': True,
  'reason': None,
  'inputs': {
    u'ansible_playbook_name': u'find_files2',
    u'ansible_parameters': u'age=-2d; host_names=192.168.56.3,192.168.56.4;
path=/usr/local/bin/'
  },
  'metrics': {
    'package': 'fn-ansible',
    'timestamp': '2019-06-19 11:06:39',
    'package_version': '1.0.0',
    'host': 'xxx.cambridge.ibm.com',
    'version': '1.0',
    'execution_time_ms': 12549
  },
  'content': {
    u'192.168.56.4': {
      'detail': 'fatal: [192.168.56.4]: UNREACHABLE! => {"changed": false, "msg": "timed out",
"unreachable": true}',
      'summary': 'failed'
    },
    u'192.168.56.3': {
      'detail': '',
      'summary': 'failed'
    }
  },
  'raw': '{"192.168.56.4": {"detail": "fatal: [192.168.56.4]: UNREACHABLE! => {\\"changed\\":
false, \\"msg\\": \\"timed out\\", \\"unreachable\\": true}", "summary": "failed"}, "192.168.56.3":
{"detail": "", "summary": "failed"}}'
}
```

The following Post-Processing Script cleans up the payload and adds it as an incident note:

```
if len(results['content'].keys()) == 0:
  note = u"No results returned on parameters: {}".format(results['inputs'])
else:
  for item in results['content']:
    note = u"{} - {}\n{}".format(item, results['inputs'], str(results['content'][item]['detail']))

incident.addNote(helper.createPlainText(note))
```

For very large data results, it may not be practical to save the results as a Note. Instead, the fn_utilities function Utilities: String to Attachment can be added to your workflow to send your Ansible results to an attachment. In this case, workflow properties are used to retain the results of this function for use by downstream functions.

Considerations

- Only the ansible-runner synchronous capability is supported.
- Playbook and artifact names must not contain unicode characters. This is a limitation in the ansible-runner package (<= 2.8.1) which should be resolved in a future release.
- The app.config setting artifact_retention_num is available to clean up previous execution files older than the specified days. Use a value of 0 to specify no deletion.
- Consider using Rule Activity Field select lists for Ansible module name and parameter restrictions. This ensures that only specific commands are used for ad-hoc executions.

- Also consider using Rule Activity Field select lists for Ansible playbook names for similar reasons to ensure only specific playbooks are supported through Resilient.
- The workflow associated with a module or playbook function will remain blocked until all host executions are complete and the results are returned. The ansible-runner Asynchronous operation corrects this restriction but remains a future enhancement.
- Playbooks should use the debug statement to return findings back to Resilient. The example below runs the find module, returning the file found using the debug statement.

```
- hosts: ""
  tasks:
  - name: Recursively find files
    find:
      paths: ""
      age: ""
      recurse: yes
      pattern: '*'
    register: files_matched

  - debug:
      msg: ""
```