ansible-runner-role Documentation

Release 1.2.1

Vladimir Botka

Jul 15, 2020

TABLE OF CONTENTS:

This role and the documentation is work in progess. Please feel free to share your feedback and report issues. Contributions are welcome.

CHAPTER

ONE

USER'S GUIDE

Table of Contents

1.1 Introduction

Run this role to install and configure Ansible Runner. Optionaly configure cron to periodically run Ansible playbooks.

- Ansible role: ansible_runner
- Supported systems:
  - FreeBSD Supported Production Releases
  - Ubuntu Supported Releases
- (Requirements in future releases: ansible_lib)

Note:

- The utility *ansible-runner* is not part of standard Ansible installation. See Installing Ansible Runner

---

See also:

- Ansible Runner documentation
- Ansible Runner source code
- REST API ansible-runner-service

1.2 Installation

The most convenient way how to install an Ansible role is to use Ansible Galaxy CLI `ansible-galaxy`. The utility comes with the standard Ansible package and provides the user with a simple interface to the Ansible Galaxy's services. For example, take a look at the current status of the role

```
shell> ansible-galaxy info vbotka.ansible_runner
```

and install it

```
shell> ansible-galaxy install vbotka.ansible_runner
```

Install the library of tasks (for future releases)

```
shell> ansible-galaxy install vbotka.ansible_lib
```

See also:

- To install specific versions from various sources see Installing content.
- Take a look at other roles `shell> ansible-galaxy search --author=vbotka`

1.3 Playbook

Below is a simple playbook that calls this role at a single host srv.example.com (2)

```
shell> cat ansible-runner.yml
- hosts: srv.example.com
  gather_facts: true
  connection: ssh
  remote_user: admin
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - vbotka.ansible_runner
```

Note: `gather_facts: true` (3) must be set to gather facts needed to evaluate OS-specific options of the role. For example to install packages the variable `ansible_os_family` is needed to select the appropriate Ansible module.

See also:

- For details see Connection Plugins (4-5)
- See also Understanding Privilege Escalation (6-8)

1.4 Debug

To see additional debug information enable debug output in the configuration

```
ar_debug: true
```

, or set the extra variable in the command

```
shell> ansible-playbook ansible-runner.yml -e 'ar_debug=true'
```

Note: The debug output of this role is optimized for the **yaml** callback plugin. Set this plugin for example in the environment `shell> export ANSIBLE_STDOUT_CALLBACK=yaml`.

See also:

- Playbook Debugger

1.5 Tags

The tags provide the user with a very useful tool to run selected tasks of the role. To see what tags are available list the tags of the role with the command

```
shell> ansible-playbook ansible-runner.yml --list-tags

playbook: ansible-runner.yml

play #1 (srv.example.com): srv.example.com TAGS: []
   TASK TAGS: [always, ar_config, ar_debug, ar_links, ar_packages, ar_vars]
```

For example, display the list of the variables and their values with the tag `ar_debug` (when the debug is enabled `ar_debug: true`)

```
shell> ansible-playbook ansible-runner.yml -t ar_debug
```

See what packages will be installed

```
shell> ansible-playbook ansible-runner.yml -t ar_packages --check
```

Install packages and exit the play

```
shell> ansible-playbook ansible-runner.yml -t ar_packages
```

1.6 Tasks

Test single tasks at single remote host *test_01*. Create a playbook

```
shell> cat ansible.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.ansible_runner
```

Customize configuration in `host_vars/test_01/ar-*.yml` and check the syntax

```
shell> ansible-playbook ansible-runner.yml --syntax-check
```

Then dry-run the selected task and see what will be changed. Replace <tag> with valid tag.

```
shell> ansible-playbook ansible-runner.yml -t <tag> --check --diff
```

When all seems to be ready run the command. Run the command twice and make sure the playbook and the configuration is idempotent

```
shell> ansible-playbook ansible-runner.yml -t <tag>
```

1.6.1 Ansible Runner packages

Synopsis

`ansible-runner` can be installed by `pip` or from distribution's packages, and ports.

See also:

- Annotated Source code *packages.yml*

Example 1: Install ansible-runner in Ubuntu by pip for admin

Create a playbook

```
shell> ansible-runner.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.ansible_runner
```

Create *host_vars/test_01/ansible-runner.yml*

```
shell> cat host_vars/test_01/ansible-runner.yml
ar_install: false
ar_pip_install: true
ar_debug: false
ar_owner: admin
```

Install ansible-runner

```
shell> ansible-playbook ansible-runner.yml -e "ar_install=true"
...
TASK [vbotka.ansible_runner : packages: Install Ansible Runner pip packages for␣
↪admin]
ok: [test_01] => (item={'name': 'ansible-runner'})
```

Show ansible-runner package installed by pip for admin

```
shell> whoami
admin

shell> pip list | grep ansible-runner
ansible-runner               1.4.6
```

Example 2: Install ansible-runner in FreeBSD from port

Create a playbook

```
shell> ansible-runner.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.ansible_runner
```

Create *host_vars/test_01/ansible-runner.yml*

```
shell> cat host_vars/test_01/ansible-runner.yml
<TBD>
```

Install ansible-runner

```
shell> ansible-playbook ansible-runner.yml -e "ar_install=true"
<TBD>
...
TASK [vbotka.ansible_runner : packages: Install Ansible Runner pip packages for␣
↪admin]
ok: [test_01] => (item={'name': 'ansible-runner'})
```

Show ansible-runner package installed by pip for admin

```
shell> whoami
 <TBD>

shell> which ansible-runner
ansible-runner                1.4.6
```

Example 1: Install ansible-runner in Ubuntu by pip for admin

Create a playbook

```
shell> ansible-runner.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.ansible_runner
```

Create *host_vars/test_01/ansible-runner.yml*

```
 shell> cat host_vars/test_01/ansible-runner.yml
 ar_install: false
 ar_pip_install: true
 ar_debug: false
 ar_owner: admin
```

Install ansible-runner

```
 shell> ansible-playbook ansible-runner.yml -e "ar_install=true"
 ...
 TASK [vbotka.ansible_runner : packages: Install Ansible Runner pip packages for␣
↪admin]
 ok: [test_01] => (item={'name': 'ansible-runner'})
```

Show ansible-runner package installed by pip for admin

```
shell> whoami
admin

shell> pip list | grep ansible-runner
ansible-runner                1.4.6
```

Example 2: Install ansible-runner in FreeBSD from port

Create a playbook

```
shell> ansible-runner.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.ansible_runner
```

Create *host_vars/test_01/ansible-runner.yml*

```
 shell> cat host_vars/test_01/ansible-runner.yml
 <TBD>
```

Install ansible-runner

```
shell> ansible-playbook ansible-runner.yml -e "ar_install=true"
<TBD>
...
TASK [vbotka.ansible_runner : packages: Install Ansible Runner pip packages for␣
↪admin]
ok: [test_01] => (item={'name': 'ansible-runner'})
```

Show ansible-runner package installed by pip for admin

```
shell> whoami
 <TBD>

shell> which ansible-runner
ansible-runner                1.4.6
```

1.7 Variables

In this chapter we describe role's default variables stored in the directory **defaults**.

See also:

- Ansible variable precedence: Where should I put a variable?

1.8 Default variables

<TBD>

[defaults/main.yml]

```
---
# defaults ansible_runner

ar_install: true
ar_debug: false
ar_backup_conf: false

# Install distro packages or pip
# false - distro packages, true -  pip
# ar_pip_install: false                 # OS specific variable see vars/defaults

# FreeBSD
freebsd_install_retries: 10
freebsd_install_delay: 5
freebsd_install_method: "packages"
# freebsd_install_method: "ports"
freebsd_use_packages: true

# Linux
linux_install_retries: 10
linux_install_delay: 5

# Python
```

(continues on next page)

(continued from previous page)

```
pip_install_retries: 10
pip_install_delay: 5
# pip_extraagrs: ""                       # Optional

# pip package dependent
# ar_pip_executable: "pip3"               # OS specific variable see vars/defaults
# ar_pip_requirements: []                 # Optional

# Configuration
ar_config: []

# Links
# ar_links: []  # OS specific variable see vars/defaults

# EOF
...
```

1.9 Best practice

Display the variables for debug if needed. Then disable this task `ar_debug: false` to speedup the playbook

```
shell> ansible-playbook ansible-runner.yml -t ar_debug
```

Install packages Then disable this task `ar_install: false` to speedup the playbook

```
shell> ansible-playbook ansible-runner.yml -t ar_packages \
                                           -e 'ar_install=true'
```

The role and the configuration data in the examples are idempotent. Once the installation and configuration have passed there should be no changes reported by *ansible-playbook* when running the playbook repeatedly. Disable debug, and install to speedup the playbook

```
shell> ansible-playbook ansible-runner.yml
```

1.10 Ansible Runner Usage Examples

1.10.1 Cron

Example 1: Run Ansible playbooks in cron

- *Run ssh-agent*
- *Run gpg-agent*
- *Wrapper ansible-runner*
- *Command for cron*
- *Crontab*

- *Email sent by cron*
- *Project*
- *Playbook*
- *Artifacts*

Run ssh-agent

`ssh-agent` is needed to provide the ssh connection plugin with the password to the private key, when used. The script below is executed by the command interpreter for login shells

```
cntrlr> cat /home/admin/.profile
if [ -n "$BASH_VERSION" ]; then
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
    if [ -f "$HOME/.bashrc_ssh" ]; then
        . "$HOME/.bashrc_ssh"
    fi
fi
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

and will start `ssh-agent` on login and prepare *SSH_ENV* (5)

```
cntrlr> cat /home/admin/.bashrc_ssh
SSH_ENV="$HOME/.ssh/environment"
function start_agent {
    echo "Initialising new SSH agent..."
    /usr/bin/ssh-agent | sed 's/^echo/#echo/' > "${SSH_ENV}"
    echo succeeded
    chmod 600 "${SSH_ENV}"
    . "${SSH_ENV}" > /dev/null
    /usr/bin/ssh-add;
}
if [ -f "${SSH_ENV}" ]; then
    . "${SSH_ENV}" > /dev/null
    #ps ${SSH_AGENT_PID} doesn't work under cywgin
    ps -ef | grep ${SSH_AGENT_PID} | grep ssh-agent$ > /dev/null || {
        start_agent;
    }
else
    start_agent;
fi
```

Example of *.ssh/environment* created by *ssh-agent*

```
cntrlr> cat /home/admin/.ssh/environment
SSH_AUTH_SOCK=/tmp/ssh-8fUkZ7qOzVPs/agent.5214; export SSH_AUTH_SOCK;
SSH_AGENT_PID=5216; export SSH_AGENT_PID;
#echo Agent pid 5216;
```

See also:

- Start ssh-agent on login - stackoverflow.com
- SSH Quick-Start Guide - FreeBSD handbook
- Single Sign-On using SSH - ssh.com

Run gpg-agent

Start gpg-agent manually by running `gpg`. `gpg-agent` is needed to provide `gpg` with the password to the private gpg key, when used. For example, to sign or encrypt emails, or to configure user's passwords with help of the `passwordstore`. The configuration below enables `gpg-agent` also within a ssh session. In particular, *no-grab* (2) allows cut&paste, *no-allow-external-cache* (3) disables any keyrings and *pinentry-curses* (4) asks for the password in the terminal instead of default *pinentry* asking in the remote (in the case of ssh) desktop window. The time to live *ttl* (5,6) is set to 24 hours. This way, it's not necessary to re-enter the password when the `cron`, which invokes the play with `gpg-agent`, is run daily.

```
cntrlr> cat ~/.gnupg/gpg-agent.conf
no-grab
no-allow-external-cache
pinentry-program /usr/bin/pinentry-curses
default-cache-ttl 86400
max-cache-ttl 86400
```

See also:

- Ansible role linux_postinstall task gpg.yml

Wrapper ansible-runner

Wrapper of *ansible-runner* will source *.ssh/environment* (14) and run the *playbook* from the *project* (15)

```
cntrlr> cat /home/admin/bin/arwrapper.bash
#!/bin/bash

runner=$HOME/bin/ansible-runner
project=$HOME/.ansible/runner/$2
playbook=${3:-all.yml}

case "$1" in
    test)
        echo $(date '+%Y-%m-%d %H:%M:%S') $runner run $project -p $playbook
        ;;
    run)
        echo $(date '+%Y-%m-%d %H:%M:%S') $0
        source $HOME/.ssh/environment
        $runner run $project -p $playbook
        ;;
    clean)
        rm -rf $project/artifacts
        ;;
    *)
        printf "$0: run|clean|test project [playbook]\n"
        exit 1
        ;;
esac
exit
```

Command for cron

The script below will use *arwrapper.bash* (5) to run the playbook *pb-01.yml* in the projects *test_01, test_02,* and *test_03* (11-13). If the command (18) succeeds the script will print *[OK]* report (23). If you don't want to receive email on success remove this line. Optionally enable/disable the cleaning of the artifacts (24).

```
cntrlr> cat /home/admin/bin/ansible-cron-test.bash
#!/bin/bash

marker=$(printf "%80s" | sed "s/ /./g")
cmd=$HOME/bin/arwrapper.bash
subcmd=${1:-run}
rc=0

typeset -A projects
projects=(
    [test_01]="pb-01.yml"
    [test_02]="pb-01.yml"
    [test_03]="pb-01.yml"
)

for project in "${!projects[@]}"; do
    for playbook in ${projects[$project]}; do
        out=$("$cmd" "$subcmd" "$project" "$playbook" 2>&1)
        if [ "$?" -eq "0" ]; then
            if [ "$subcmd" = "test" ]; then
                printf "[DRY] $out\n"
            fi
            printf "[OK]  "$project" "$playbook" PASSED\n"
            $cmd clean $project
        else
            printf "[ERR] $out\n$marker\n"
            rc=1
        fi
    done
 done
 exit $rc
```

Crontab

Schedule the script in *cron*

```
cntrlr> whoami
admin
cntrlr> crontab -l
MAILTO=admin
#Ansible: Ansible runner daily test
50 20 * * * $HOME/bin/ansible-cron-test.bash
```

See also:

- Ansible role's task FreeBSD postinstall cron.yml
- Ansible role's task Linux postinstall cron.yml

Email sent by cron

In our case the */etc/aliases* redirect the emails for *root* to the user *admin*. Cron will report the result of the scpript *ansible-cron-test.bash*. If you want to receive email on a failure only remove the *[OK]* report from the script and optionally clean the *artifacts*. The *artifacts* will be available for a review if the script fails

```
Date: Tue,  7 Jul 2020 20:50:06 +0200 (CEST)
From: Cron Daemon <root@cntrlr.example.com>
To: admin@cntrlr.example.com
Subject: Cron <admin@cntrlr> $HOME/bin/ansible-cron-test.bash

[OK]  test_01 pb-01.yml PASSED
[OK]  test_02 pb-01.yml PASSED
[OK]  test_03 pb-01.yml PASSED
```

Project

Example of the project's directory without the artifacts. The artifacts will be created by *ansible-runner*

```
cntrlr> tree /home/admin/.ansible/runner/test_01
/home/admin/.ansible/runner/test_01
├── env
├── inventory
│   └── hosts
└── project
    ├── ansible.cfg
    ├── group_vars
    ├── host_vars
    └── pb-01.yml
```

Note: It's necesary to provide *ansible-playbook* wiht the *vault password* if any data were encrypted. Use env/cmdline. For example

---

```
cntrl> cat /home/admin/.ansible/runner/test_01/env/cmdline
--vault-password-file $HOME/.vault-psswd
```

See also:

- Runner Input Directory Hierarchy
- Example playbook how to create projects pb_create_runner_private.yml

Playbook

Example of a playbook used in the test

```
cntrlr> cat /home/admin/.ansible/runner/test_01/project/pb-01.yml
- hosts: test_01
  remote_user: admin
  gather_facts: no
  tasks:
    - debug:
        msg: TEST
```

Artifacts

Example of the project's artifacts

```
cntrl> tree /home/admin/.ansible/runner/test_01/artifacts/
/home/admin/.ansible/runner/test_01/artifacts
└── aaa5d36e-e8d4-432a-ab52-b69062c85311
    ├── command
    ├── fact_cache
    ├── job_events
    │   ├── 1-2b5c9412-f0c4-45dc-a425-5c8c29e37ec0.json
    │   ├── 2-5ce0c5a2-1f02-cdab-8869-00000000001f.json
    │   ├── 3-5ce0c5a2-1f02-cdab-8869-000000000021.json
    │   ├── 4-28749e27-409a-46c4-9551-7ce80c02be83.json
    │   ├── 5-997d90c1-6357-45c6-8df9-437c2940c74e.json
    │   └── 6-6e41cf27-8c1e-4266-9ffb-8a54375bd4cc.json
    ├── rc
    ├── status
    └── stdout
```

See also:

- Runner Artifacts Directory Hierarchy
- ansible_lib: al_runner_events

1.10.2 Job events

Example 1: List artifacts' job events

Test negative result

Let's modify the playbook so that it'll fail. For example (8)

```
cntrlr> cat ~/.ansible/runner/test_02/project/pb-01.yml
- hosts: test_02
  remote_user: admin
  gather_facts: no
  tasks:
    - debug:
        msg: TEST
    - command: /usr/bin/false
```

Cron email on failure

Then the cron task in the example *Cron: Example 1* will fail and *admin* will receive an email similar to this one

```
Date: Wed,  8 Jul 2020 13:27:07 +0200 (CEST)
From: Cron Daemon <root@cntrlr.example.com>
To: admin@cntrlr.example.com
Subject: Cron <admin@cntrlr> $HOME/bin/ansible-cron-test.bash

[OK]  test_01 pb-01.yml PASSED
[ERR] 2020-07-08 13:27:03 /home/admin/bin/arwrapper.bash

PLAY [test_02] ***************************************************************

TASK [debug] *****************************************************************
ok: [test_02] => {
    "msg": "TEST"
    }

    TASK [command] ***********************************************************
    fatal: [test_02.g2.netng.org]: FAILED! =>
    {"changed": true,
     "cmd": ["/usr/bin/false"],
     "delta": "0:00:00.013809",
     "end": "2020-07-08 +13:26:32.197207",
     "msg": "non-zero return code",
     "rc": 1,
     "start": "2020-07-08 13:26:32.183398",
     "stderr": "",
     "stderr_lines": [],
     "stdout": "",
     "stdout_lines": []}

    PLAY RECAP *******************************************************************
    test_02: ok=1 changed=0 unreachable=0 failed=1 skipped=0 rescued=0 ignored=0
    ................................................................................
    [OK]  test_03 pb-01.yml PASSED
```

Artifacts

Let's take look at the artifacts of the failed project

```
cntrlr> tree ~/.ansible/runner/test_02/artifacts/
/home/admin/.ansible/runner/test_02/artifacts/
└── 0428ede5-40c2-48f9-b33d-b9d1a64609af
    ├── command
    ├── fact_cache
    ├── job_events
    │   ├── 1-ff7d4af5-26bc-4d4c-8eac-6c75e547cb22.json
    │   ├── 2-5ce0c5a2-1f02-fda4-7a07-00000000001f.json
    │   ├── 3-5ce0c5a2-1f02-fda4-7a07-000000000021.json
    │   ├── 4-a1e17955-d452-424d-a1c1-bb4b387fd180.json
    │   ├── 5-97175f4b-9c82-4160-a17c-32a3e6d0c3ff.json
    │   ├── 6-5ce0c5a2-1f02-fda4-7a07-000000000022.json
    │   ├── 7-e1a3349e-199f-4ad7-969c-8680bbb1bac0.json
    │   ├── 8-bb64ec8e-d1b0-4114-9093-9bbd6807b293.json
```

(continues on next page)

(continued from previous page)

```
    │   └── 9-72588652-8937-4eda-9aa7-b6bc443e4aa9.json
    ├── rc
    ├── status
    └── stdout

3 directories, 13 files
```

Playbook

Prepare a playbook to help with the analysis of the artifacts. For example, the playbook below will use Ansible library task al_runner_events.yml (13) and display selected attributes (18) from the *job events*. Feel free to modify *msg* (18) and display other attributes

```
cntrlr> cat ar-events.yml
- hosts: localhost
  gather_facts: false

  vars:
    my_home: "{{ lookup('env','HOME') }}"
    al_runner_events_dir: "{{ my_home ~
    '/.ansible/runner/test_02/artifacts/0428ede5-40c2-48f9-b33d-b9d1a64609af/job_
↪events' }}"

  tasks:
    - include_role:
        name: vbotka.ansible_lib
        tasks_from: al_runner_events
        apply:
          tags: always
      tags: always
    - debug:
        msg: "{{ item.counter }} {{ item.event }}"
      loop: "{{ al_runner_events_list|sort(attribute='counter') }}"
      loop_control:
        label: "{{ item.counter }}"
      tags: events
    - debug:
        msg: "{{ item.stdout }}"
      loop: "{{ al_runner_events_list|sort(attribute='counter') }}"
      loop_control:
        label: "{{ item.counter }}"
      when: item.event == 'runner_on_failed'
      tags: failed
```

See also:

- Examples of ansible-runner

Events

The play below gives the list of the events

```
cntrlr> ansible-playbook ar-events.yml -t events | grep msg\":
    "msg": "1 playbook_on_start"
    "msg": "2 playbook_on_play_start"
    "msg": "3 playbook_on_task_start"
    "msg": "4 runner_on_start"
    "msg": "5 runner_on_ok"
    "msg": "6 playbook_on_task_start"
    "msg": "7 runner_on_start"
    "msg": "8 runner_on_failed"
    "msg": "9 playbook_on_stats"
```

Failed event(s)

The next play displays the details of the failed event(s)

```
cntrlr> echo -e $(ansible-playbook ar-events.yml -t failed | grep msg\":)
    "msg": "fatal: [test_02]: FAILED! =>{
    \"changed\": true,
    \"cmd\": [\"/usr/bin/false\"],
    \"delta\": \"0:00:00.014716\",
    \"end\": \"2020-07-08 17:05:56.104764\",
    \"msg\": \"non-zero return code\",
    \"rc\": 1,
    \"start\": \"2020-07-08 17:05:56.090048\",
    \"stderr\": \"\",
    \"stderr_lines\": [],
    \"stdout\": \"\",
    \"stdout_lines\": []}"
```

1.11 Troubleshooting

1.11.1 Commented issues

There are reported issues at GitHub. Some of them influence *ansible-runner* in an unexpected and undocumented ways. See the commented issues to avoid unnecesary confusion and investigation looking for the difference between a bug and a feature. Check with the links to see the issues' current status.

- *ANSIBLE_CALLBACK_PLUGINS in envvars is not working. #219*

ANSIBLE_CALLBACK_PLUGINS in envvars is not working. #219

- Updated: July,9 2020
- Status: Closed without fix Aug 2019

It's not possbile to change Ansible callback plugin. `ansible-runner` ignores environment variable `ANSIBLE_STDOUT_CALLBACK` and uses hardwired callback plugins (2,4). See runner_config.py

```
if 'AD_HOC_COMMAND_ID' in self.env:
    self.env['ANSIBLE_STDOUT_CALLBACK'] = 'minimal'
else:
    self.env['ANSIBLE_STDOUT_CALLBACK'] = 'awx_display'
```

To test it, set the enironment variables of the project *test_02*

```
shell> cat ~/.ansible/runner/test_02/env/envvars
---
MY_TEST_VAR: my-test-var
ANSIBLE_STDOUT_CALLBACK: actionable
```

Prepare a playbook

```
shell> cat pb-02.yml
- hosts: test_02
  remote_user: admin
  gather_facts: true
  tasks:
    - debug:
        msg: "HOME [{{ lookup('env','HOME') }}]"
    - debug:
        msg: "MY_TEST_VAR [{{ lookup('env','MY_TEST_VAR') }}]"
    - debug:
        msg: "ANSIBLE_STDOUT_CALLBACK [{{ lookup('env','ANSIBLE_STDOUT_CALLBACK') }}]"
```

and test it

```
shell> ansible-runner --version
1.4.6

shell> ansible-runner run ~/.ansible/runner/test_02 -p pb-02.yml

TASK [debug] *******************************************************************
ok: [test_02] => {
    "msg": "HOME [/home/admin]"
}

TASK [debug] *******************************************************************
ok: [test_02] => {
    "msg": "MY_TEST_VAR [my-test-var]"
}

TASK [debug] *******************************************************************
ok: [test_02] => {
    "msg": "ANSIBLE_STDOUT_CALLBACK [awx_display]"
}
```

CHAPTER

TWO

ANNOTATED SOURCE CODE

Table of Contents

2.1 Tasks

2.1.1 main.yml

Synopsis: Tasks of the playbook.

Description of the task.

[main.yml]

```
---
# tasks for ansible_runner

- import_tasks: vars.yml
  tags: [ar_vars, always]

- import_tasks: debug.yml
  when: ar_debug|bool
  tags: [ar_debug, always]

- import_tasks: packages.yml
  when: ar_install|bool
  tags: ar_packages

```

(continues on next page)

(continued from previous page)

```
- import_tasks: links.yml
  tags: ar_links

- import_tasks: config.yml
  tags: ar_config

# EOF
...
```

2.1.2 config.yml

Synopsis: Configure config.

Description of the task.

[config.yml]

```
---

- name: "config: configure ansible-runner"
  debug:
    msg: No config
  when: ar_debug|bool

# EOF
...
```

2.1.3 debug.yml

Synopsis: Configure debug.

Description of the task.

[debug.yml]

```
---

- name: "debug: Ansible Runner"
  vars:
    msg: |
      ansible_os_family [{{ ansible_os_family }}]
      ansible_distribution [{{ ansible_distribution }}]
      ansible_distribution_major_version [{{ ansible_distribution_major_version }}]
      ansible_distribution_version [{{ ansible_distribution_version }}]
      ansible_distribution_release [{{ ansible_distribution_release }}]
      ansible_python_version [{{ ansible_python_version }}]

      ar_install [{{ ar_install }}]

      freebsd_install_method [{{ freebsd_install_method }}]
      freebsd_use_packages [{{ freebsd_use_packages }}]
      freebsd_install_retries [{{ freebsd_install_retries }}]
      freebsd_install_delay [{{ freebsd_install_delay }}]

      linux_install_retries [{{ linux_install_retries }}]
```

(continues on next page)

(continued from previous page)

```
      linux_install_delay [{{ linux_install_delay }}]

      pip_install_retries [{{ pip_install_retries }}]
      pip_install_delay [{{ pip_install_delay }}]
      pip_extraagrs [{{ pip_extraagrs|default('UNDEFINED') }}]

      ar_pip_install [{{ ar_pip_install }}]
      ar_pip_executable [{{ ar_pip_executable }}]
      ar_pip_requirements [{{ pip_requirements|default('UNDEFINED') }}]

      ansible_user_id [{{ ansible_user_id }}]

      ar_owner [{{ ar_owner }}]
      ar_backup_conf [{{ ar_backup_conf }}]

      ar_packages
      {{ ar_packages|to_nice_yaml }}
      ar_links
      {{ ar_links|to_nice_yaml }}

      ar_config
      {{ ar_config|to_nice_yaml }}

  debug:
    msg: "{{ msg.split('\n') }}"

# EOF
...
```

2.1.4 links.yml

Synopsis: Configure links.

Description of the task.

[links.yml]

```
---

- name: "links: Create directories for links"
  file:
    state: "directory"
    dest: "{{ item.dest|dirname }}"
  loop: "{{ ar_links }}"

- name: "links: Create links"
  file:
    state: "link"
    src: "{{ item.src }}"
    dest: "{{ item.dest }}"
    force: "{{ item.force|default(false) }}"
  loop: "{{ ar_links }}"

# EOF
...
```

2.1.5 packages.yml

Synopsis: Configure packages.

Description of the task.

[packages.yml]

```
---

# packages ------------------------------------------------------------

# FreeBSD
- name: "packages: Install Ansible Runner FreeBSD packages"
  block:
    - name: "packages: Install Ansible Runner packages FreeBSD"
      pkgng:
        name: "{{ item.name }}"
      loop: "{{ ar_packages }}"
      register: result
      until: result is succeeded
      retries: "{{ freebsd_install_retries }}"
      delay: "{{ freebsd_install_delay }}"
    - name: "packages: Debug FreeBSD packages"
      when: ar_debug|bool
      debug:
        var: result
  when:
    - not ar_pip_install
    - ansible_os_family == "FreeBSD"
    - freebsd_install_method|lower == "packages"

- name: "packages: Install FreeBSD ports"
  block:
    - name: "packages: Install Ansible Runner ports FreeBSD"
      portinstall:
        name: "{{ item.name }}"
        use_packages: "{{ freebsd_use_packages }}"
      loop: "{{ ar_packages }}"
      register: result
      until: result is succeeded
      retries: "{{ freebsd_install_retries }}"
      delay: "{{ freebsd_install_delay }}"
    - name: "packages: Debug FreeBSD ports"
      when: ar_debug|bool
      debug:
        var: result
  when:
    - not ar_pip_install
    - ansible_os_family == "FreeBSD"
    - freebsd_install_method|lower == "ports"

# Linux
- name: "packages: Install Ansible Runner packages Linux"
  block:
    - name: "packages: Install Ansible Runner packages Linux"
      package:
        name: "{{ item.name }}"
      loop: "{{ ar_packages }}"
```

(continues on next page)

(continued from previous page)

```
      register: result
      until: result is succeeded
      retries: "{{ linux_install_retries }}"
      delay: "{{ linux_install_delay }}"
    - name: "packages: Debug Linux"
      when: ar_debug|bool
      debug:
        var: result
  when:
    - not ar_pip_install
    - ansible_os_family == "RedHat" or ansible_os_family == "Debian"

# pip -----------------------------------------------------------------
- name: "packages: Test {{ ar_pip_executable }} exists"
  when: ar_pip_install
  block:
    - name: "packages: Stat {{ ar_pip_executable }}"
      stat:
        path: "{{ ar_pip_executable }}"
      register: result
    - name: "packages: Not exists {{ ar_pip_executable }}"
      fail:
        msg: "[ERROR] {{ ar_pip_executable }} does not exist."
      when: not result.stat.exists

- name: "packages: Install Ansible Runner pip packages for {{ ar_owner }}"
  when: ar_pip_install
  become_user: "{{ ar_owner }}"
  become: true
  changed_when: false  # Note 1.
  pip:
    name: "{{ item.name }}"
    executable: "{{ ar_pip_executable }}"
    version: "{{ item.version|default(omit) }}"
    state: "{{ item.state|default(omit) }}"
    extra_args: "{{ pip_extraagrs|default(omit) }}"
  loop: "{{ ar_packages }}"
  register: result
  until: result is succeeded
  retries: "{{ pip_install_retries }}"
  delay: "{{ pip_install_delay }}"

- name: "packages: Debug pip packages"
  when:
    - ar_pip_install
    - ar_debug|bool
  debug:
    var: result

- name: "packages: Install Ansible Runner pip requirements for {{ ar_owner }}"
  when:
    - ar_pip_install
    - ar_pip_requirements is defined
  become_user: "{{ ar_owner }}"
  become: true
  changed_when: false  # Note 1.
  pip:
```

(continues on next page)

(continued from previous page)

```
    requirements: "{{ ar_pip_requirements }}"
    executable: "{{ ar_pip_executable }}"
    extra_args: "{{ pip_extraagrs|default(omit) }}"
  register: result
  until: result is succeeded
  retries: "{{ pip_install_retries }}"
  delay: "{{ pip_install_delay }}"

- name: "packages: Debug pip requirements"
  when:
    - ar_pip_install
    - ar_debug|bool
  debug:
    var: result

# Note 1.
# The pip module isn't always idempotent #28952
# https://github.com/ansible/ansible/issues/28952

# EOF
...
```

2.1.6 vars.yml

Synopsis: Configure vars.

Description of the task.

[vars.yml]

```
---

- name: "Declare ar_owner when undefined"
  when: ar_owner is undefined
  set_fact:
    ar_owner: "{{ ansible_user_id }}"

- name: "Default vars for {{ ansible_os_family }}
                          {{ ansible_distribution }}
                          {{ ansible_distribution_release }}"
  include_vars: "{{ item }}"
  with_first_found:
    - files:
        - "{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml"
        - "{{ ansible_distribution }}.yml"
        - "{{ ansible_os_family }}.yml"
        - "defaults.yml"
        - "default.yml"
      paths: "{{ role_path }}/vars/defaults"

- name: "Custom vars for {{ ansible_os_family }}
                         {{ ansible_distribution }}
                         {{ ansible_distribution_release }}"
  include_vars: "{{ item }}"
  with_first_found:
    - files:
```

(continues on next page)

(continued from previous page)

```
        - "{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml"
        - "{{ ansible_distribution }}.yml"
        - "{{ ansible_os_family }}.yml"
        - "defaults.yml"
        - "default.yml"
      paths: "{{ role_path }}/vars"

# EOF
...
```

CHAPTER

THREE

COPYRIGHT

Redistribution and use in source (RST DocUtils) and 'compiled' forms (XML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (RST Docutils) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Important: THIS DOCUMENTATION IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROVIDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER

FOUR

LEGAL NOTICE

All product names, logos, and brands are property of their respective owners.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search