ara Documentation

Release 0.16.6.dev2

Red Hat

Sep 12, 2019

Contents

CHAPTER 1

Table of Contents

1.1 FAQ

1.1.1 What is ARA ?

ARA makes Ansible runs easier to visualize, understand and troubleshoot.

ARA provides four things:

1. An *Ansible callback plugin* to record playbook runs into a local or remote database
2. The *ara_record* and *ara_read* pair of Ansible modules to record and read persistent data with ARA
3. A *CLI client* to query the database
4. A *dynamic, database-driven web interface* that can also be *generated and served from static files*

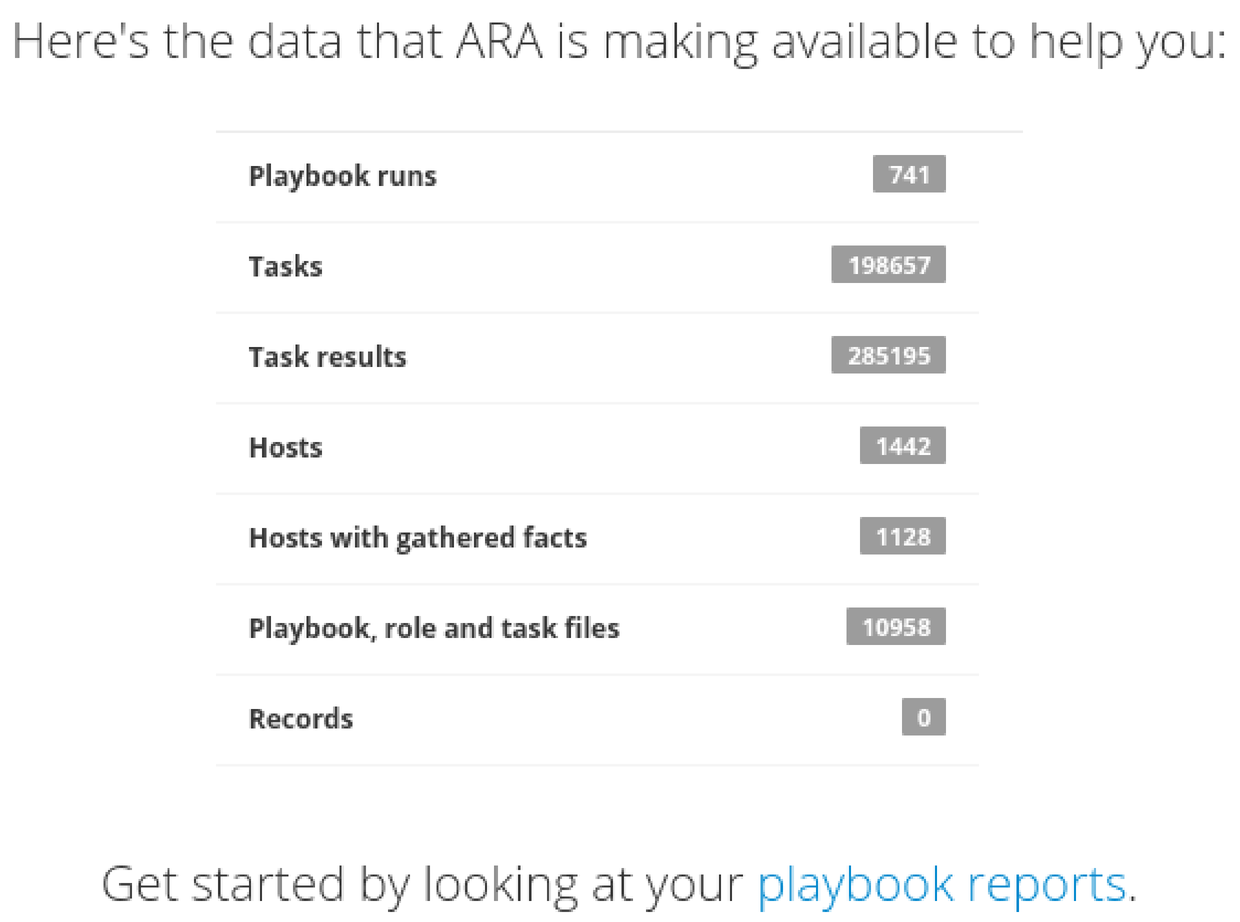
1.1.2 What does the web interface look like ?

A video preview and explanation of the web interface is available on YouTube, featuring playbook runs from the OpenStack-Ansible project.

Otherwise, here's some screenshots highlighting some of ARA's features:

Home page

The home page highlights the data recorded by ARA:



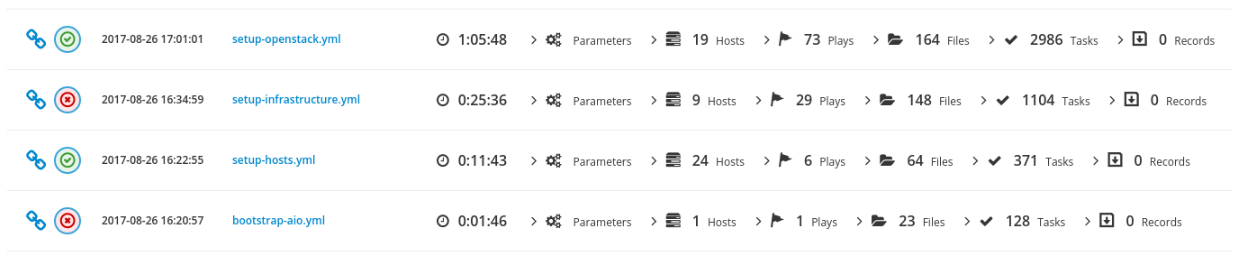
Here's the data that ARA is making available to help you:

Playbook runs	741
Tasks	198657
Task results	285195
Hosts	1442
Hosts with gathered facts	1128
Playbook, role and task files	10958
Records	0

Get started by looking at your playbook reports.

Playbook reports

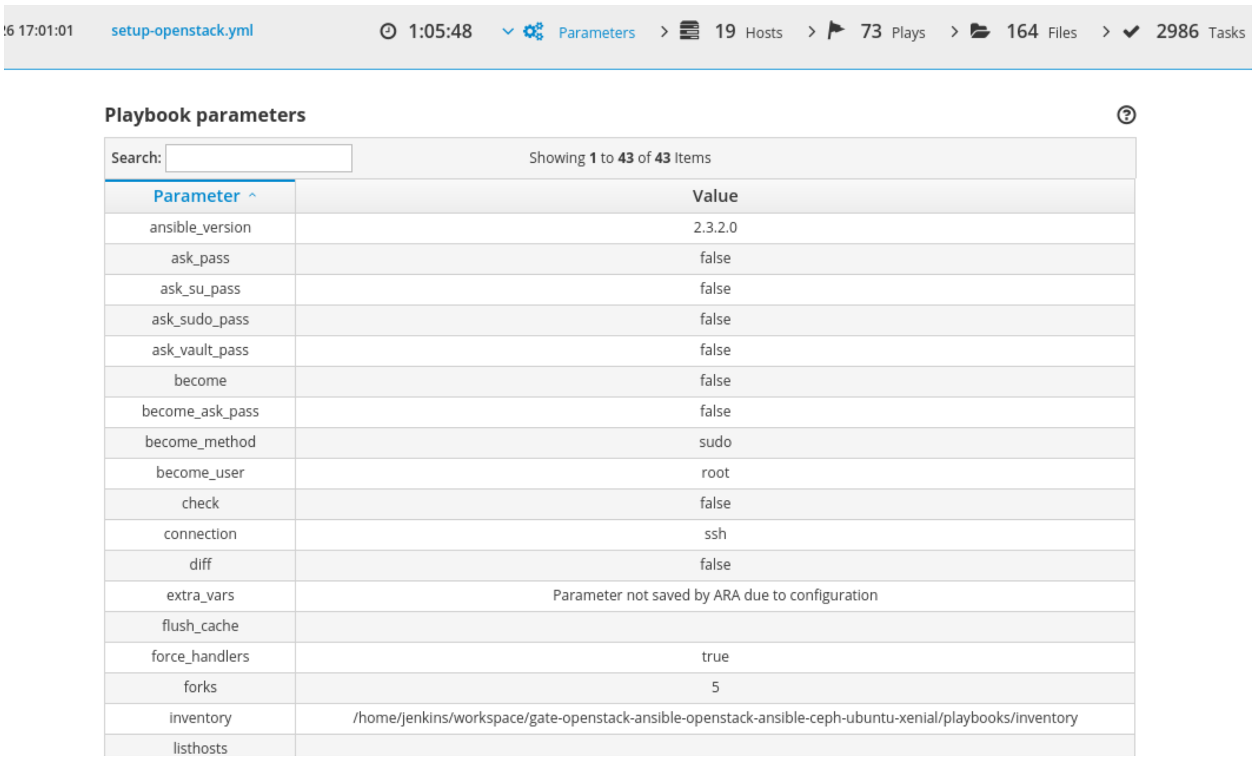
The core of the web application interface revolves around one and single page where you'll be able to find all the information about your playbooks:



2017-08-26 17:01:01	setup-openstack.yml	1:05:48	Parameters	19 Hosts	73 Plays	164 Files	2986 Tasks	0 Records
2017-08-26 16:34:59	setup-infrastructure.yml	0:25:36	Parameters	9 Hosts	29 Plays	148 Files	1104 Tasks	0 Records
2017-08-26 16:22:55	setup-hosts.yml	0:11:43	Parameters	24 Hosts	6 Plays	64 Files	371 Tasks	0 Records
2017-08-26 16:20:57	bootstrap-aio.yml	0:01:46	Parameters	1 Hosts	1 Plays	23 Files	128 Tasks	0 Records

Ansible parameters

ARA stores parameters and options passed to your ansible-playbook command:



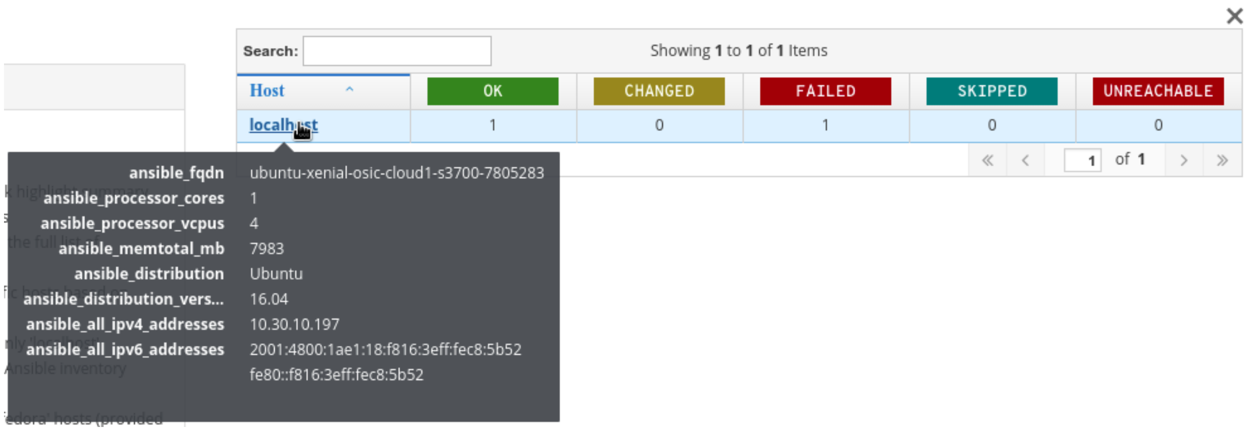
Playbook parameters

Search: Showing 1 to 43 of 43 Items

Parameter	Value
ansible_version	2.3.2.0
ask_pass	false
ask_su_pass	false
ask_sudo_pass	false
ask_vault_pass	false
become	false
become_ask_pass	false
become_method	sudo
become_user	root
check	false
connection	ssh
diff	false
extra_vars	Parameter not saved by ARA due to configuration
flush_cache	
force_handlers	true
forks	5
inventory	/home/jenkins/workspace/gate-openstack-ansible-openstack-ansible-ceph-ubuntu-xenial/playbooks/inventory
listhosts	

Playbook host summary

Quickly have a glance at summary statistics or host facts for your playbook:



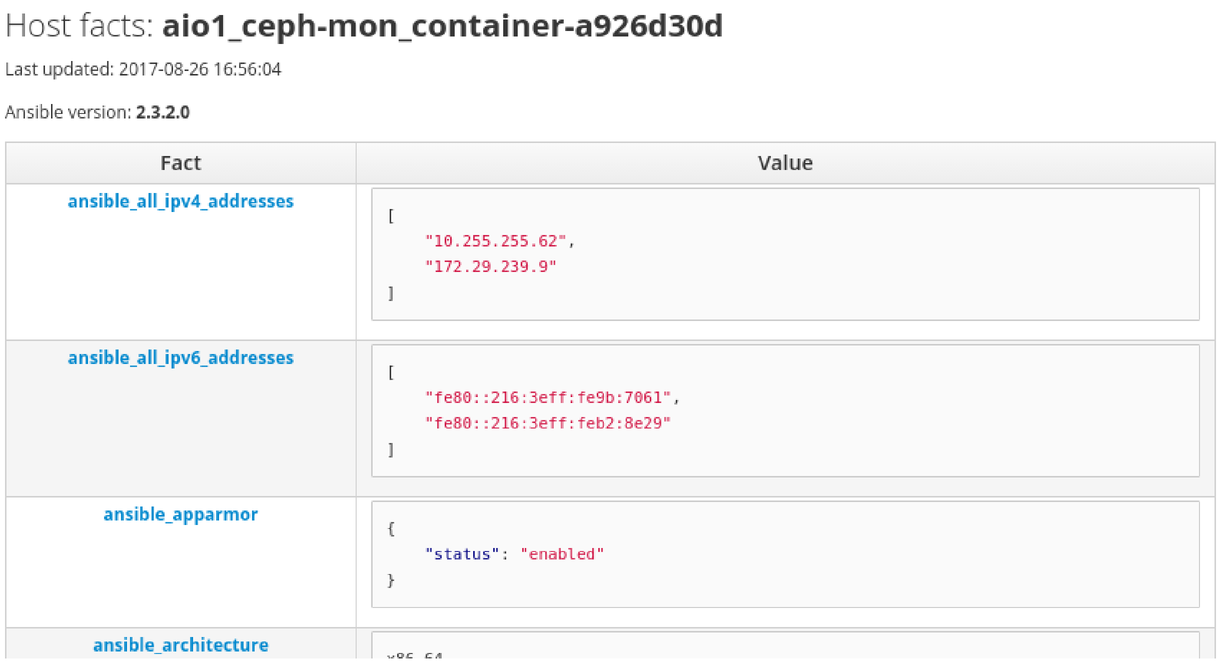
Search: Showing 1 to 1 of 1 Items

Host	OK	CHANGED	FAILED	SKIPPED	UNREACHABLE
localhost	1	0	1	0	0

ansible_fqdn	ubuntu-xenial-osic-cloud1-s3700-7805283
ansible_processor_cores	1
ansible_processor_vcpus	4
ansible_memtotal_mb	7983
ansible_distribution	Ubuntu
ansible_distribution_vers...	16.04
ansible_all_ipv4_addresses	10.30.10.197
ansible_all_ipv6_addresses	2001:4800:1ae1:18:f816:3eff:fec8:5b52 fe80::f816:3eff:fec8:5b52

Recorded host facts

If Ansible gathered facts as part of your playbook, ARA will save them and make them available:



Host facts: **aio1_ceph-mon_container-a926d30d**

Last updated: 2017-08-26 16:56:04

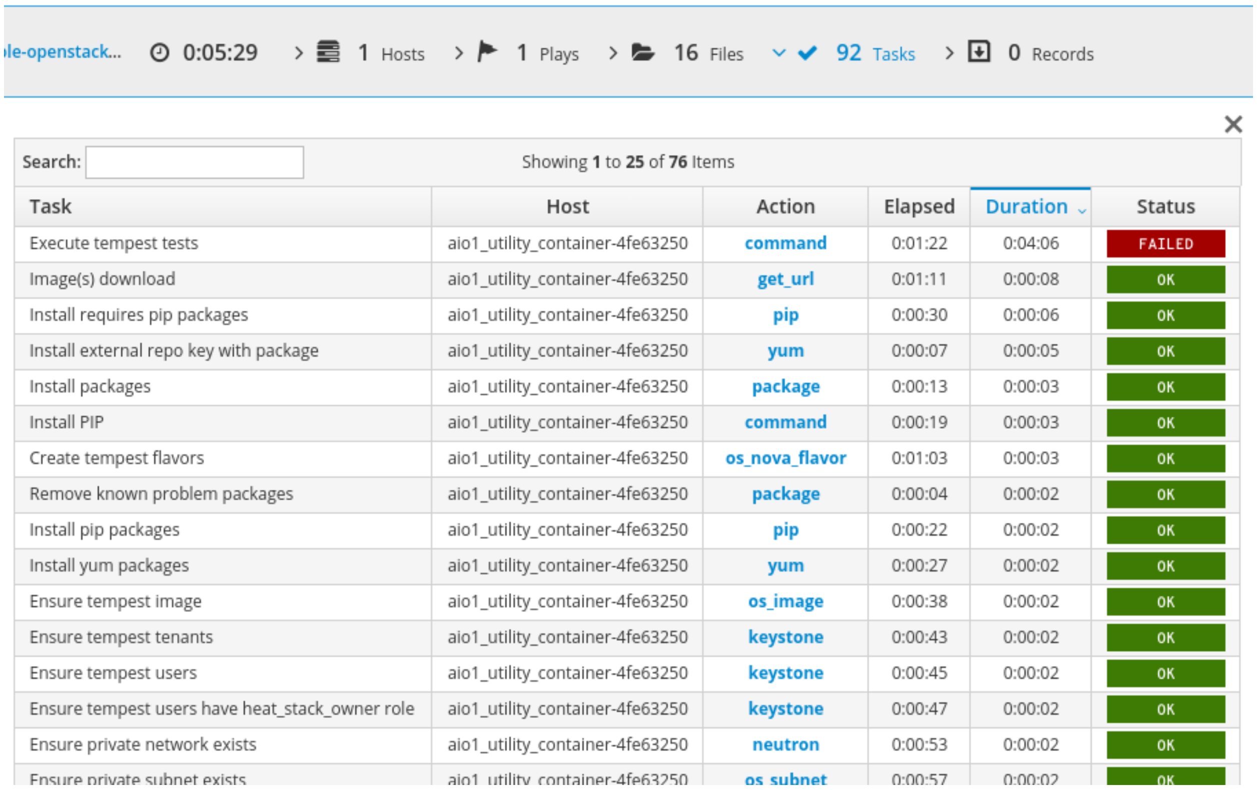
Ansible version: **2.3.2.0**

Fact	Value
ansible_all_ipv4_addresses	`[ "10.255.255.62", "172.29.239.9" ]`
ansible_all_ipv6_addresses	`[ "fe80::216:3eff:fe9b:7061", "fe80::216:3eff:feb2:8e29" ]`
ansible_apparmor	`{ "status": "enabled" }`
ansible_architecture	x86_64

Organized task results

Quickly and easily get insight into your task results.

Sort them by duration to find which took the longest time

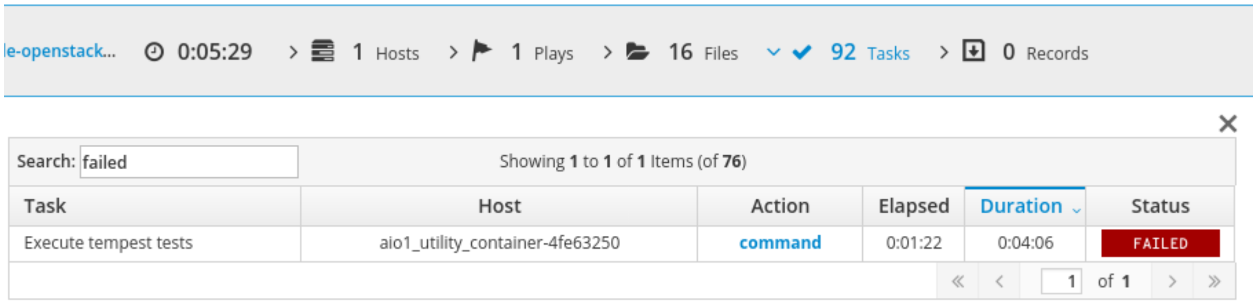


ble-openstack... 0:05:29 1 Hosts 1 Plays 16 Files 92 Tasks 0 Records

Search: Showing **1** to **25** of **76** Items

Task	Host	Action	Elapsed	Duration	Status
Execute tempest tests	aio1_utility_container-4fe63250	command	0:01:22	0:04:06	FAILED
Image(s) download	aio1_utility_container-4fe63250	get_url	0:01:11	0:00:08	OK
Install requires pip packages	aio1_utility_container-4fe63250	pip	0:00:30	0:00:06	OK
Install external repo key with package	aio1_utility_container-4fe63250	yum	0:00:07	0:00:05	OK
Install packages	aio1_utility_container-4fe63250	package	0:00:13	0:00:03	OK
Install PIP	aio1_utility_container-4fe63250	command	0:00:19	0:00:03	OK
Create tempest flavors	aio1_utility_container-4fe63250	os_nova_flavor	0:01:03	0:00:03	OK
Remove known problem packages	aio1_utility_container-4fe63250	package	0:00:04	0:00:02	OK
Install pip packages	aio1_utility_container-4fe63250	pip	0:00:22	0:00:02	OK
Install yum packages	aio1_utility_container-4fe63250	yum	0:00:27	0:00:02	OK
Ensure tempest image	aio1_utility_container-4fe63250	os_image	0:00:38	0:00:02	OK
Ensure tempest tenants	aio1_utility_container-4fe63250	keystone	0:00:43	0:00:02	OK
Ensure tempest users	aio1_utility_container-4fe63250	keystone	0:00:45	0:00:02	OK
Ensure tempest users have heat_stack_owner role	aio1_utility_container-4fe63250	keystone	0:00:47	0:00:02	OK
Ensure private network exists	aio1_utility_container-4fe63250	neutron	0:00:53	0:00:02	OK
Ensure private subnet exists	aio1_utility_container-4fe63250	os_subnet	0:00:57	0:00:02	OK

Search and filter by task name, host, action or status

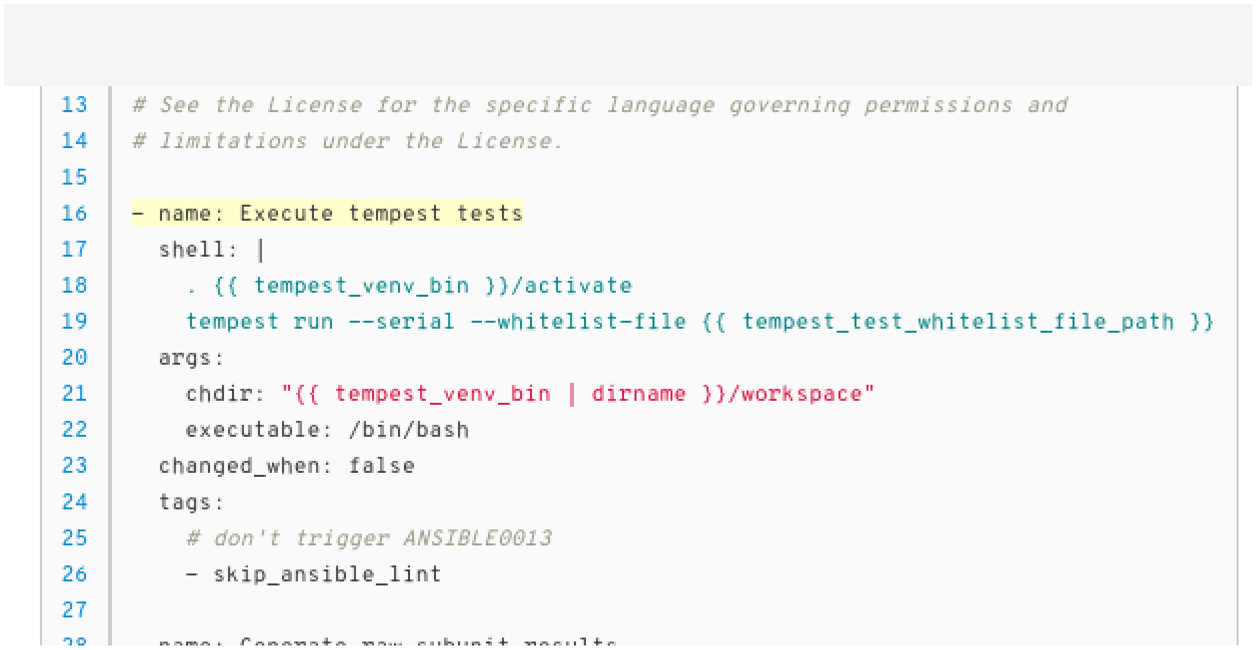


Search: failed

Showing 1 to 1 of 1 Items (of 76)

Task	Host	Action	Elapsed	Duration	Status
Execute tempest tests	aio1_utility_container-4fe63250	command	0:01:22	0:04:06	FAILED

Click on the action to get context on where a specific task ran

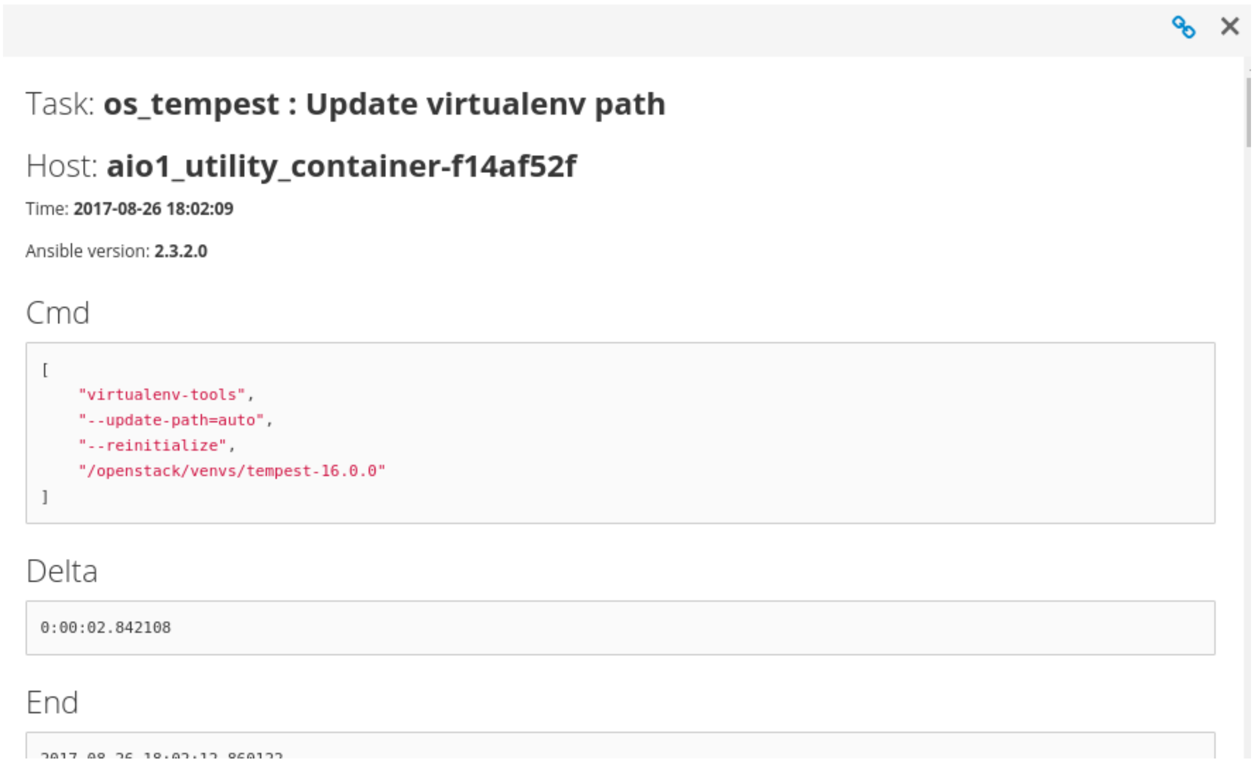


```
# See the License for the specific language governing permissions and
# limitations under the License.

- name: Execute tempest tests
  shell: |
    . {{ tempest_venv_bin }}/activate
    tempest run --serial --whitelist-file {{ tempest_test_whitelist_file_path }}
  args:
    chdir: "{{ tempest_venv_bin | dirname }}/workspace"
    executable: /bin/bash
  changed_when: false
  tags:
    # don't trigger ANSIBLE0013
    - skip_ansible_lint

```

Click on the status to dig into all the data made available by Ansible


Task: **os_tempest : Update virtualenv path**

Host: **aio1_utility_container-f14af52f**

Time: **2017-08-26 18:02:09**

Ansible version: **2.3.2.0**

Cmd

```
[
    "virtualenv-tools",
    "--update-path=auto",
    "--reinitialize",
    "/openstack/venvs/tempest-16.0.0"
]
```

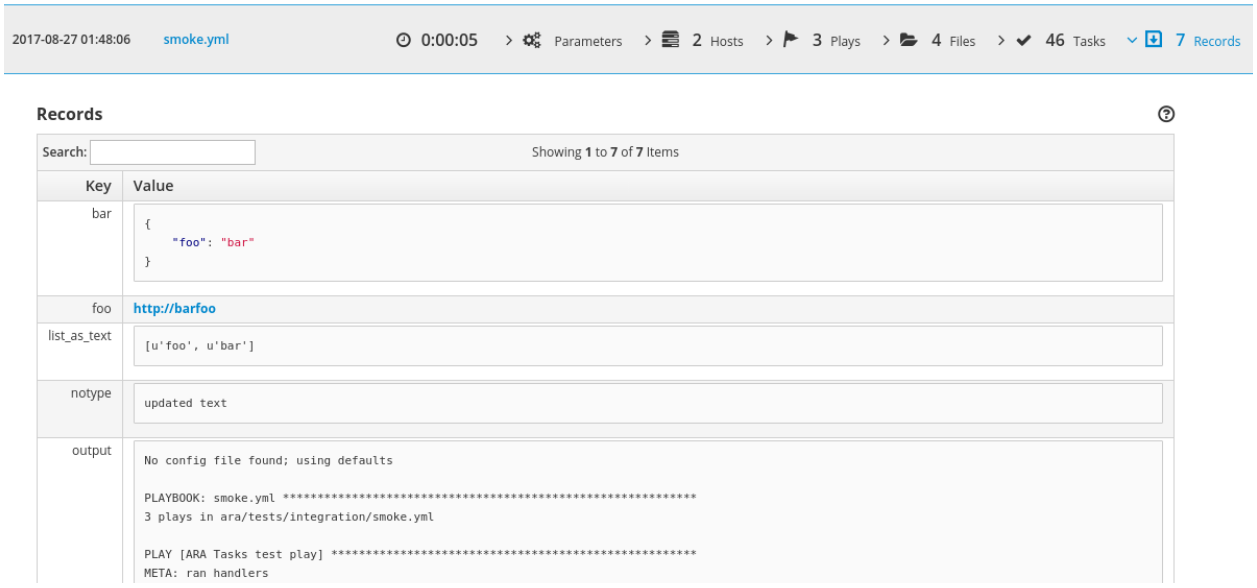
Delta

```
0:00:02.842108
```

End

Arbitrarily recorded data

The *ara_record* and *ara_read* built-in Ansible modules allow you to write and read arbitrary data, making them available in the web interface:



Records

Search:

Showing 1 to 7 of 7 Items

Key	Value
bar	{ "foo": "bar" }
foo	http://barfoo
list_as_text	[u'foo', u'bar']
notype	updated text
output	No config file found; using defaults ...

```
No config file found; using defaults

PLAYBOOK: smoke.yml ************************************************************
3 plays in ara/tests/integration/smoke.yml

PLAY [ARA Tasks test play] *****************************************************
META: ran handlers
```

1.1.3 What versions of Ansible are supported ?

The upstream Ansible community and maintainers provide support for the latest two major stable releases and ARA follows the same support cycle.

At this time, the minimum required version of Ansible to run the latest version of ARA is **2.4.1.0**. New development is tested against the latest versions of **2.4**, **2.5** as well as `devel` which is currently the future version of Ansible, **2.6**.

If you are using a release of Ansible that is no longer supported, we strongly encourage you to upgrade as soon as possible in order to benefit from the latest features and security fixes.

Older unsupported versions of Ansible can contain unfixed security vulnerabilities (*CVE*).

1.1.4 Does ARA support running on Python 3 ?

Yes.

The support for running ARA on a python 3 environment landed in ARA 0.14.0. Previous versions would not work on python 3.

1.1.5 What's an Ansible callback ?

Ansible Callbacks are essentially hooks provided by Ansible. Ansible will send an event and you can react to it with a callback. You could use a callback to do things like print additional details or, in the case of ARA, record the playbook run data in a database.

1.1.6 Why is ARA being developed ?

Ansible is an awesome tool. It can be used for a lot of things.

Reading and interpreting the output of an `ansible-playbook` run, especially one that is either long running, involves a lot of hosts or prints a lot of output can be tedious. This is especially true when you happen to be running Ansible hundreds of times during the day, through automated means – for example when doing continuous integration or continuous delivery.

ARA aims to do one thing and do it well: Record Ansible runs and provide means to visualize these records to help you be more efficient.

1.1.7 Why don't you use Ansible Tower (AWX), Rundeck or Semaphore ?

Ansible Tower is a product from Red Hat while Ansible AWX is the upstream open source version of Tower. ARA is not mutually exclusive with either: you can use it with your Tower or AWX deployment but it's only job is to provide reporting.

Ansible Tower, AWX, Semaphore and Rundeck all have something in common. They are tools that control (or want to control) the whole workflow from end-to-end and they do so in a fairly "centralized" fashion where everything runs from the place where the software is hosted.

They provide features like inventory management, ACLs, playbook execution, editing features and so on.

Since they are the ones actually running Ansible, it makes sense that they can record and display the data in an organized way.

ARA is decentralized and self-contained: `pip install ara`, configure the callback in `ansible.cfg`, run a playbook and it'll be recorded, wherever it is. ARA doesn't want to do things like inventory management, provide editing features or control the workflow. It just wants to record data and provide an intuitive interface for it.

When using ARA, you can store and browse your data locally and this is in fact the default behavior. You are not required to use a central server or upload your data elsewhere.

While the features provided by Tower and other products are definitely nice, the scope of ARA is kept narrow on purpose. By doing so, ARA remains a relatively simple application that is very easy to install and configure. It does not require any changes to your setup or workflow, it adds itself in transparently and seamlessly.

For more information regarding the core values and the scope for the ARA project, refer to the project *manifesto*.

1.1.8 Can Ansible with ARA run on a different server than the web application ?

ARA comes bundled in an all-in-one package: callback, modules, web application and command line interface. When you install ARA, you get all of those out of the box.

The ARA components themselves are mostly decoupled, however, and as long as they can all communicate with the same database, you'll get the same experience.

You can run Ansible with ARA on your laptop, save to a local sqlite database and run the web application from the embedded server, everything offline, if that's what you need.

However, you can also, for example, use a *MySQL configuration* to have Ansible and ARA send data to a remote database server instead.

Another server could host the web application with *Apache+mod_wsgi* with the same database configuration and you would be accessing the same recorded data.

You could also have ARA installed on yet another computer with the same configuration and the command line interface will be able to retrieve the data automatically as well.

1.1.9 Can ARA be used outside the context of OpenStack or continuous integration ?

ARA has no dependencies or requirements with OpenStack or Jenkins for CI. You can use ARA with Ansible for any playbook in any context.

ARA is completely generic but was developed out of necessity to make troubleshooting OpenStack continuous integration jobs faster and easier.

1.2 Installing ARA

Installing ARA is easy.

1.2.1 RHEL, CentOS, Fedora packages

Required dependencies

```
yum install gcc python-devel libffi-devel openssl-devel redhat-rpm-config
```

Development or integration testing dependencies

```
yum install python-setuptools libselinux-python libxml2-devel libxslt-devel
easy_install pip
pip install tox
```

1.2.2 Ubuntu, Debian packages

Required dependencies

```
apt-get install gcc python-dev libffi-dev libssl-dev
```

Development or integration testing dependencies

```
apt-get install python-pip libxml2-dev libxslt1-dev
pip install tox
```

1.2.3 Installing ARA from trunk source

```
pip install git+https://git.openstack.org/openstack/ara
```

1.2.4 Installing ARA from latest release on PyPi

```
pip install [--user] ara
```

When installing ARA using `--user`, command line scripts will be installed inside `~/.local/bin` folder which may not be in `PATH`. You may want to assure that this folder is in PATH or to use the alternative calling method `python -m ara` which calls Ansible module directly.

The alternative calling method has the advantage that allows user to control which python interpreter would be used. For example you could install ARA in both python2 and python3 and call the one you want.

1.3 Configuration

1.3.1 Ansible

To begin using ARA, you'll first need to set up Ansible so it knows about the the ARA *callback* and, if necessary, the *ara_record* and *ara_read* modules.

The callback and modules are bundled when installing ARA but you need to know where they have been installed in order to let Ansible know where they are located.

This location will be different depending on your operating system, how you are installing ARA and whether you are using Python 2 or Python 3.

ARA ships a set of convenience Python modules to help you configure Ansible to use it.

They can be used like so:

```
$ python -m ara.setup.path
/usr/lib/python2.7/site-packages/ara

$ python -m ara.setup.action_plugins
/usr/lib/python2.7/site-packages/ara/plugins/actions

```

(continues on next page)

(continued from previous page)

```
$ python -m ara.setup.callback_plugins
/usr/lib/python2.7/site-packages/ara/plugins/callbacks
```

Using ansible.cfg

This sets up a new ansible.cfg file to load the callbacks and modules from the appropriate locations:

```
$ python -m ara.setup.ansible | tee ansible.cfg
[defaults]
callback_plugins=/usr/lib/python2.7/site-packages/ara/plugins/callbacks
action_plugins=/usr/lib/python2.7/site-packages/ara/plugins/actions
```

Or alternatively, if you have a customized ansible.cfg file, you can retrieve only what you need using the other helpers such as the following:

- `python -m ara.setup.callback_plugins`
- `python -m ara.setup.action_plugins`

Using environment variables

Depending on the context and your use case, configuring Ansible using environment variables instead of an `ansible.cfg` file might be more convenient.

ARA provides a helper module that prints out the necessary export commands:

```
$ python -m ara.setup.env
export ANSIBLE_CALLBACK_PLUGINS=/usr/lib/python2.7/site-packages/ara/plugins/callbacks
export ANSIBLE_ACTION_PLUGINS=/usr/lib/python2.7/site-packages/ara/plugins/actions
```

Note that the module doesn't actually run those exports, you'll want to run them yourself, add them in a bash script or a bashrc, etc.

1.3.2 ARA

ARA uses the same mechanism and configuration files as Ansible to retrieve it's configuration. It comes with sane defaults that can be customized if need be.

The order of priority is the following:

1. Environment variables
2. `./ansible.cfg` (*In the current working directory*)
3. `~/.ansible.cfg` (*In the home directory*)
4. `/etc/ansible/ansible.cfg`

When using the ansible.cfg file, the configuration options must be set under the ara namespace, as follows:

```
[ara]
variable = value
```

Note: The callback, CLI client and web application all share the same settings. For example, if you configure the database location, all three will use that location.

1.3.3 Parameters and their defaults

Environment variable	[ara] ansible.cfg variable	Default value
ARA_DIR	dir	~/.ara
ARA_DATABASE	database	sqlite:///~/.ara/ansible.sqlite
ARA_HOST	host	127.0.0.1
ARA_PORT	port	9191
ARA_APPLICATION_ROOT	application_root	/
ARA_LOG_CONFIG	logconfig	None
ARA_LOG_FILE	logfile	~/.ara/ara.log
ARA_LOG_LEVEL	loglevel	INFO
ARA_LOG_FORMAT	logformat	%(asctime)s - %(levelname)s - %(message)s
ARA_IGNORE_FACTS	ignore_facts	ansible_env
ARA_IGNORE_PARAMETERS	ignore_parameters	extra_vars
ARA_IGNORE_EMPTY_GENERATION	ignore_empty_generation	True
ARA_IGNORE_MIMETYPE_WARNINGS	ignore_mimetype_warnings	True
ARA_PLAYBOOK_OVERRIDE	playbook_override	None
ARA_PLAYBOOK_PER_PAGE	playbook_per_page	10
ARA_RESULT_PER_PAGE	result_per_page	25
SQLALCHEMY_ECHO	sqlalchemy_echo	False
SQLALCHEMY_POOL_SIZE	sqlalchemy_pool_size	None (default managed by flask-sqlalchemy)
SQLALCHEMY_POOL_TIMEOUT	sqlalchemy_pool_timeout	None (default managed by flask-sqlalchemy)
SQLALCHEMY_POOL_RECYCLE	sqlalchemy_pool_recycle	None (default managed by flask-sqlalchemy)

ARA_DIR

Base directory where ARA will store it's log file and sqlite database, unless specified otherwise.

ARA_DATABASE

ARA records Ansible data in a database. The callback, the CLI client and the web application all need to know where that database is located.

ARA ensures the database exists and it's schema is created when it is run.

ARA comes out of the box with sqlite enabled and no additional setup required. If, for example, you'd like to use MySQL instead, you will need to create a database and it's credentials:

```
CREATE DATABASE ara;
CREATE USER ara@localhost IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON ara.* TO ara@localhost;
FLUSH PRIVILEGES;
```

And then setup the database connection:

```
export ARA_DATABASE="mysql+pymysql://ara:password@localhost/ara"
# or
[ara]
database = mysql+pymysql://ara:password@localhost/ara
```

When using a different database driver such as MySQL (pymysql), you also need to make sure you install the driver:

```
# From pypi
pip install pymysql
# For RHEL derivatives
yum install python-PyMySQL
# For Debian or Ubuntu
apt-get install python-pymysql
```

Alternatively, if you prefer PostgreSQL, you can do the following in psql:

```
CREATE ROLE ara WITH LOGIN PASSWORD 'password';
CREATE DATABASE ara OWNER ara;
GRANT ALL ON DATABASE ara TO ara;
```

Be sure you update your pg_hba.conf afterwards if needed.

Then, setup the database connection:

```
export ARA_DATABASE="postgresql+psycopg2://ara:password@localhost:5432/ara"
# or
[ara]
database = postgresql+psycopg2://ara:password@localhost:5432/ara
```

You will need to install the database driver by:

```
# From pypi
pip install psycopg2
# For RHEL derivatives
yum install python-psycopg2
# For Debian or Ubuntu
apt-get install python-psycopg2
```

ARA_HOST

The host on which the development server will bind to by default when using the `ara-manage runserver` command.

It is equivalent to the `-h` or `--host` argument of the `ara-manage runserver` command.

ARA_PORT

The port on which the development server will listen on by default when using the `ara-manage runserver` command.

It is equivalent to the `-p` or `--port` argument of the `ara-manage runserver` command.

ARA_APPLICATION_ROOT

The path at which the web application should be loaded.

The default behavior is to load the application at the root (`/`) of your host. Change this parameter if you'd like to host your application elsewhere.

For example, `/ara` would make the application available under `http://host/ara` instead of `http://host/`.

ARA_LOG_CONFIG

Path to a python logging config file.

If the filename ends in `.yaml` or `.yml` the file will be loaded as yaml. If the filename ends in `.json` the file will be loaded as json. The resulting dict for either will be treated as a logging config dict and passed to *logging.config.dictConfig*.

Otherwise it will be assumed to a logging config file and the path will be passed to *logging.config.fileConfig*.

If this option is given it superseeds the other individual log options.

ARA_LOG_FILE

Path to the logfile to store ARA logs in.

ARA_LOG_LEVEL

The loglevel to adjust debug or verbosity.

ARA_LOG_FORMAT

The log format of the logs.

ARA_IGNORE_FACTS

When Ansible gathers host facts or uses the setup module, your host facts are recorded by ARA and are also available as part of your reports.

By default, only the host fact `ansible_env` is not saved due to the sensitivity of the information it could contain such as tokens, passwords or otherwise privileged information.

This configuration allows you to customize what ARA will and will not save. It is a list, provided by comma-separated values.

ARA_IGNORE_PARAMETERS

ARA will, by default, save every parameter and option passed to ansible-playbook (except `extra-vars`) and make them available as part of your reports.

If, for example, you use extra_vars to send a password or secret variable to your playbooks, it is likely you don't want this saved in ARA's database.

This configuration allows you to customize what ARA will and will not save. It is a list, provided by comma-separated values.

ARA_IGNORE_EMPTY_GENERATION

When using `ara generate html`, whether or not to ignore warnings provided by flask-frozen about endpoints for which the application found no available data.

For example, if you do not use the `ara_record` module as part of your playbooks, this avoids printing a *MissingURLGeneratorWarning* because there is no recorded data to render.

ARA_IGNORE_MIMETYPE_WARNINGS

When using `ara generate html`, whether or not to ignore file mimetype warnings provided by flask-frozen.

ARA_PLAYBOOK_OVERRIDE

This configuration is exposed mostly for the purposes of the `ara generate html` and `ara generate junit` commands but you can use it as well.

ARA_PLAYBOOK_OVERRIDE will limit the playbooks displayed in the web application to the list of playbook IDs specified. This is expected to be playbook IDs (ex: retrieved through `ara playbook list`) in a comma-separated list.

ARA_PLAYBOOK_PER_PAGE

This is the amount of playbooks runs shown in a single page in the ARA web interface. The default is `10` but you might want to tweak this number up or down depending on the amount of hosts, tasks and task results contained in your playbooks. This directly influences the weight of the pages that will end up being displayed. Setting this value too high might yield very heavy pages.

Set this parameter to `0` to disable playbook listing pagination entirely.

ARA_RESULT_PER_PAGE

This is the amount of results shown in a single page in the different data tables such as hosts, plays and tasks of the ARA web interface. The default is `25` but you might want to tweak this number up or down depending on your preference. This has no direct impact on the weight of the page being sent for the reports as these data tables are rendered on the client side.

Set this parameter to `0` to disable pagination for results entirely.

1.3.4 The CLI client and the web application

The CLI client and the web application do not need to be run on the same machine that Ansible is executed from but they do need a database and know it's location.

Both could query a local sqlite database or a remote MySQL database, for example.

1.4 Web Server Configuration

The web interface provided by ARA is a simple Flask application. There are many ways to deploy and host a Flask application, here we cover two different ways which should help you get started.

In any case, ARA will need to be installed before you proceed. Refer to the *documentation* if you need to know how to install ARA.

1.4.1 Embedded server

ARA comes bundled with an embedded server meant for development or debugging purposes.

Note that any serious deployment should probably not be running off of this as it is not meant to be serving clients directly at any kind of scale.

To start the development server, use the provided `ara-manage runserver` command:

```
$ ara-manage runserver --help
usage: ara-manage runserver [-?] [-h HOST] [-p PORT] [--threaded]
                            [--processes PROCESSES] [--passthrough-errors]
                            [-d] [-D] [-r] [-R]

Runs the Flask development server i.e. app.run()

optional arguments:
  -?, --help            show this help message and exit
  -h HOST, --host HOST
  -p PORT, --port PORT
  --threaded
  --processes PROCESSES
  --passthrough-errors
  -d, --debug           enable the Werkzeug debugger (DO NOT use in production
                        code)
  -D, --no-debug        disable the Werkzeug debugger
  -r, --reload          monitor Python files for changes (not 100% safe for
                        production use)
  -R, --no-reload       do not monitor Python files for changes
```

To expose any non-default configurations to the development server (such as the database location), the same principles as usual apply – you need to have an *ansible.cfg file or declare environment variables*.

For example, to fire the server to listen on all IPv4 addresses on port 8080 while using a database at `/tmp/ara.sqlite`:

```
$ export ARA_DATABASE="sqlite:////tmp/ara.sqlite"
$ ara-manage runserver -h 0.0.0.0 -p 8080
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

1.4.2 Apache+mod_wsgi

Note: ARA needs to be installed on the server where Apache will be running. Refer to the *documentation* if you need to know how to install ARA.

Fedora/CentOS/RHEL

Install Apache+mod_wsgi

```
yum install httpd mod_wsgi
systemctl enable httpd
systemctl start httpd
```

Create a directory for Ansible and ARA

This directory is where we will store the files that Apache will need to read and write to.

```
mkdir -p /var/www/ara
```

Copy ARA's WSGI script to the web directory

ARA provides a WSGI script when it is installed: `ara-wsgi`. We need to copy it to the directory we just created, `/var/www/ara`.

The location where `ara-wsgi` is installed depends on how you installed ARA and the distribution you are running. You can use `which` to find where it is located:

```
cp -p $(which ara-wsgi) /var/www/ara/
```

Create the Ansible and ARA configuration

The defaults provided by ARA and Ansible are not suitable for a use case where we are deploying with Apache. We need to provide different settings:

```
cat <<EOF >/var/www/ara/ansible.cfg
[defaults]
# This directory is required to store temporary files for Ansible and ARA
local_tmp = /var/www/ara/.ansible/tmp

[ara]
# This will default the database and logs location to be inside that directory.
dir = /var/www/ara/.ara
EOF
```

For additional parameters, such as the database location or backend, look at the *configuration documentation*.

File permissions and SElinux

Make sure everything is owned by Apache so it can read and write to the directory:

```
chown -R apache:apache /var/www/ara
```

Additionally, if you are running with selinux enforcing, you need to allow Apache to manage the files in `/var/www/ara`. You can toggle the `httpd_unified` boolean for that:

```
setsebool -P httpd_unified 1
```

Apache configuration

Set up the Apache virtual host at `/etc/httpd/conf.d/ara.conf`:

```
<VirtualHost *:80>
    # Replace ServerName by your hostname
    ServerName ara.domain.tld

    ErrorLog /var/log/httpd/ara-error.log
    LogLevel warn
    CustomLog /var/log/httpd/ara-access.log combined

    WSGIDaemonProcess ara user=apache group=apache processes=4 threads=1
    WSGIScriptAlias / /var/www/ara/ara-wsgi

    SetEnv ANSIBLE_CONFIG /var/www/ara/ansible.cfg

    <Directory /var/www/ara>
        WSGIProcessGroup ara
        WSGIApplicationGroup %{GLOBAL}
        Require all granted
    </Directory>
</VirtualHost>
```

Restart Apache and you're done:

```
systemctl restart httpd
```

You should now be able to access the web interface at the domain you set up !

Debian/Ubuntu

Install Apache+mod_wsgi

```
apt-get install apache2 libapache2-mod-wsgi
systemctl enable apache2
systemctl start apache2
```

Create the directory for Ansible and ARA

This directory is where we will store the files that Apache will need to read and write to.

```
mkdir -p /var/www/ara
```

Copy ARA's WSGI script to the web directory

ARA provides a WSGI script when it is installed: `ara-wsgi`. We need to copy it to the directory we just created, `/var/www/ara`.

The location where `ara-wsgi` is installed depends on how you installed ARA and the distribution you are running. You can use `which` to find where it is located:

```
cp -p $(which ara-wsgi) /var/www/ara/
```

Create the Ansible and ARA configuration

The defaults provided by ARA and Ansible are not suitable for a use case where we are deploying with Apache. We need to provide different settings:

```
cat <<EOF >/var/www/ara/ansible.cfg
[defaults]
# This directory is required to store temporary files for Ansible and ARA
local_tmp = /var/www/ara/.ansible/tmp

[ara]
# This will default the database and logs location to be inside that directory.
dir = /var/www/ara/.ara
EOF
```

For additional parameters, such as the database location or backend, look at the *configuration documentation*.

File permissions

Make sure everything is owned by Apache so it can read and write to the directory:

```
chown -R www-data:www-data /var/www/ara
```

Apache configuration

Set up the Apache virtual host at `/etc/apache2/sites-available/ara.conf`:

```
<VirtualHost *:80>
    # Replace ServerName by your hostname
    ServerName ara.domain.tld

    ErrorLog /var/log/apache2/ara-error.log
    LogLevel warn
    CustomLog /var/log/apache2/ara-access.log combined

    WSGIDaemonProcess ara user=www-data group=www-data processes=4 threads=1
    WSGIScriptAlias / /var/www/ara/ara-wsgi

    SetEnv ANSIBLE_CONFIG /var/www/ara/ansible.cfg

    <Directory /var/www/ara>
        WSGIProcessGroup ara
        WSGIApplicationGroup %{GLOBAL}
        Require all granted
    </Directory>
</VirtualHost>
```

Ensure the configuration is enabled:

```
a2ensite ara
```

Restart Apache and you're done:

```
systemctl restart apache2
```

You should now be able to access the web interface at the domain you set up !

1.4.3 Serving static HTML reports

Nginx Configuration

Assuming that you are storing ARA reports as static html using a Nginx server you may find this configuration useful as it assures that prezipped files (like `index.html.gz`) are served transparently by the server.

```
location /artifacts {
    gzip_static on;
    root /var/www/html;
    autoindex on;
    index index.html index.htm;
    rewrite ^(.*)/$ $1/index.html;
}
```

You may need a different nginx build that has the ngx_http_gzip_static_module compiled. For example nginx from EPEL (CentOS/RHEL) yum repositories includes this module.

1.5 Serving ARA sqlite databases over http

Hosting statically generated reports is not very efficient at a large scale. The reports are relatively small in size but can contain thousands of files if you are generating a report that contains thousands of tasks.

However, using a centralized database (such as MySQL) might not be optimal either. Perhaps due to the latency or maybe because of the concurrency of the runs. It is also possible you are not interested in aggregating data in the first place and would rather keep individual reports.

ARA ships a bundled WSGI middleware, `wsgi_sqlite.py`.

This middleware allows you to store your `ansible.sqlite` databases on a web server (for example, a logserver for your CI jobs) and load these databases on the fly without needing to generate static reports.

It works by matching a requested URL (ex: `http://logserver/some/path/ara-report`) against the filesystem location (ex: `/srv/static/logs/some/path/ara-report/ansible.sqlite`) and loading ARA's web application so that it reads from the database directly.

To put this use case into perspective, it was "benchmarked" against a single job from the OpenStack-Ansible project:

- 4 playbooks
- 4647 tasks
- 4760 results
- 53 hosts, of which 39 had gathered host facts
- 416 saved files

Generating a static report from that database takes ~1min30s on an average machine. It weighs 63MB (27MB recursively gzipped), contains 5321 files and 5243 directories.

This middleware allows you to host the exact same report on your web server just by storing the sqlite database which is just one file and weighs 5.6MB.

1.5.1 wsgi_sqlite configuration

Configuration for the `wsgi_sqlite.py` script can be done through environment variables, for example with Apache's `SetEnv` directive.

ARA_WSGI_USE_VIRTUALENV

Enable virtual environment usage if ARA is installed in a virtual environment. You will need to set `ARA_WSGI_VIRTUALENV_PATH` if enabling this.

Defaults to `0`, set to `1` to enable.

ARA_WSGI_VIRTUALENV_PATH

When using a virtual environment, where the virtualenv is located. Defaults to `None`, set to the absolute path of your virtualenv.

ARA_WSGI_TMPDIR_MAX_AGE

This WSGI middleware creates temporary directories which should be discarded on a regular basis to avoid them accumulating. This is a duration, in seconds, before cleaning directories up.

Defaults to `3600`.

ARA_WSGI_LOG_ROOT

Absolute path on the filesystem that matches the `DocumentRoot` of your webserver vhost.

For a `DocumentRoot` of `/srv/static/logs`, this value should be `/srv/static/logs`.

Defaults to `/srv/static/logs`.

ARA_WSGI_DATABASE_DIRECTORY

Subdirectory in which ARA sqlite databases are expected to reside in. For example, `ara-report` would expect: `http://logserver/some/path/ara-report/ansible.sqlite`.

This variable should match the `WSGIScriptAliasMatch` pattern of your webserver vhost.

Defaults to `ara-report`.

1.5.2 Using wsgi_sqlite with Apache's mod_wsgi

The vhost requires you to redirect requests to `*/ara-report/*` to the WSGI middleware. In order to do so, the vhost must look like the following:

```
<VirtualHost *:80>
  # Remember that DocumentRoot and ARA_WSGI_LOG_ROOT must match
  DocumentRoot /srv/static/logs
  ServerName logs.domain.tld

  ErrorLog /var/log/httpd/logs.domain.tld-error.log
  LogLevel warn
  CustomLog /var/log/httpd/logs.domain.tld-access.log combined

  # Look out for the user/group which is different based on your distro
  WSGIDaemonProcess ara user=apache group=apache processes=4 threads=1

  SetEnv ARA_WSGI_TMPDIR_MAX_AGE 3600
  SetEnv ARA_WSGI_LOG_ROOT /srv/static/logs
  SetEnv ARA_WSGI_DATABASE_DIRECTORY ara-report

  <Directory "/usr/bin">
    <Files "ara-wsgi-sqlite">
      Require all granted
    </Files>
  </Directory>

  # Redirect everything after /ara-report to the middleware
  WSGIScriptAliasMatch ^.*/ara-report /usr/bin/ara-wsgi-sqlite
</VirtualHost>
```

1.5.3 Using a virtual environment

When using ARA from a virtual environment, you need to adjust your configuration accordingly.

For example, your vhost might need to look like this instead:

```
<VirtualHost *:80>
  # Remember that DocumentRoot and ARA_WSGI_LOG_ROOT must match
  DocumentRoot /srv/static/logs
  ServerName logs.domain.tld

  ErrorLog /var/log/httpd/logs.domain.tld-error.log
  LogLevel warn
  CustomLog /var/log/httpd/logs.domain.tld-access.log combined

  # Look out for the user/group which is different based on your distro
  WSGIDaemonProcess ara user=apache group=apache processes=4 threads=1 python-home=/
↪opt/venv/ara

  SetEnv ARA_WSGI_USE_VIRTUALENV 1
  SetEnv ARA_WSGI_VIRTUALENV_PATH /opt/venv/ara
  SetEnv ARA_WSGI_TMPDIR_MAX_AGE 3600
  SetEnv ARA_WSGI_LOG_ROOT /srv/static/logs
  SetEnv ARA_WSGI_DATABASE_DIRECTORY ara-report
```

(continues on next page)

(continued from previous page)

```
  <Directory "/opt/venv/ara/bin">
    <Files "ara-wsgi-sqlite">
      Require all granted
    </Files>
  </Directory>

  # Redirect everything after /ara-report to the middleware
  WSGIScriptAliasMatch ^.*/ara-report /opt/venv/ara/bin/ara-wsgi-sqlite
</VirtualHost>
```

1.6 Usage

Once ARA is *installed* and *configured*, you're ready to use it!

1.6.1 Using the callback

The callback is executed by Ansible automatically once the path is set properly in the `callback_plugins` Ansible configuration.

After running an Ansible playbook, the database will be created if it doesn't exist and will be used automatically.

1.6.2 Using the ara_record module

ARA comes with a built-in Ansible module called `ara_record`.

This module can be used as an action for a task in your Ansible playbooks in order to register whatever you'd like in a key/value format, for example:

```
---
- name: Test playbook
  hosts: localhost
  tasks:
    - name: Get git version of playbooks
      command: git rev-parse HEAD
      register: git_version

    # Registering the result of an ara_record tasks is equivalent to
    # doing an ara_read on the key
    - name: Record git version
      ara_record:
        key: "git_version"
        value: "{{ git_version.stdout }}"
      register: version

    - name: Print recorded data
      debug:
        msg: "{{ version.playbook_id}} - {{ version.key }}: {{ version.value }}
```

It also supports data types which will have an impact on how the value will be displayed in the web interface. The default type if not specified is "text". Example usage:

```
---
- ara_record:
    key: "{{ item.key }}"
    value: "{{ item.value }}"
    type: "{{ item.type }}"
  with_items:
    - { key: "log", value: "error", type: "text" }
    - { key: "website", value: "http://domain.tld", type: "url" }
    - { key: "data", value: '{ "key": "value" }', type: "json" }
    - { key: "somelist", value: ['one', 'two'], type: "list" }
    - { key: "somedict", value: {'key': 'value' }, type: "dict" }
```

It is also possible to run an `ara_record` task on a specific playbook that might already be completed. This is particularly useful for recording data that might only be available or computed after your playbook run has been completed:

```
---
# Write data to a specific (previously run) playbook
# (Retrieve playbook uuid's with 'ara playbook list')
- ara_record:
    playbook: uuuu-iiii-dddd-0000
    key: logs
    value: "{{ lookup('file', '/var/log/ansible.log') }}"
    type: text
```

Or as an ad-hoc command:

```
ansible localhost -m ara_record \
    -a "playbook=uuuu-iiii-dddd-0000 key=logs value={{ lookup('file', '/var/log/
↪ansible.log') }}"
```

This data will be recorded inside ARA's database and associated with the particular playbook run that was executed.

You can then query ARA, either through the CLI or the web interface to see the recorded values.

1.6.3 Using the ara_read module

ARA comes with a built-in Ansible module called `ara_read` that can read data that was previously recorded with `ara_record` within the same playbook run.

This module can be used as an action for a task anywhere in your in your Ansible playbooks as long as it is within the same playbook run. It can be re-used across plays or roles if necessary, for example:

```
---
- name: Test play on localhost
  hosts: localhost
  tasks:
    - name: Compute md5sum of file
      command: md5sum file
      register: local_mdfive

    - name: Record md5sum of dile
      ara_record:
        key: "md5sum"
        value: "{{ local_mdfive.stdout }}"
```

(continues on next page)

(continued from previous page)

```
- name: Test play on remote hosts
  hosts: webservers
  tasks:
      - name: Retrieve md5sum
        ara_read:
          key: "md5sum"
        register: mdfive

      - name: Compare md5sum of files
        shell: diff <(md5sum file) <(echo "{{ mdfive.value }}")
```

It is also possible to run an `ara_read` task on a specific playbook that might already be completed. This is particularly useful for reading data that might only be available or computed after your playbook run has been completed:

```
---
# Read data from a specific (previously run) playbook
# (Retrieve playbook uuid's with 'ara playbook list')
- ara_read:
    playbook: uuuu-iiii-dddd-0000
    key: logs
  register: logs
```

Or as an ad-hoc command:

```
ansible localhost -m ara_read -a "playbook=uuuu-iiii-dddd-0000 key=logs"
```

Note: `ara_read` on a specific playbook id should only be used if you need to tie data back into Ansible for other tasks. If you just need to browse or view recorded data on the command line, you should probably be using the ARA CLI: `ara data show`.

1.6.4 Looking at the data

Once you've run ansible-playbook at least once, the database will be populated with data:

```
# Example with sqlite
$ sqlite3 ~/.ara/ansible.sqlite
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
sqlite> select * from playbooks;
15d05ac3-95b6-4767-ab1e-5365f76e5b09|playbooks/test.yml|2016-05-14 03:17:57.
↪866103|2016-05-14 03:17:59.451822

# Example with MySQL
# mysql -e "select * from ara.playbooks;"
+--------------------------------------+--------------+---------------------+---------
↪------------+
| id                                   | path         | time_start          | time_
↪end            |
+--------------------------------------+--------------+---------------------+---------
↪------------+
| 48912da8-4e83-4fdb-b73d-62b03f2a5ed9 | playbook.yml | 2016-05-14 03:27:39 | 2016-05-
↪14 03:27:39 |
+--------------------------------------+--------------+---------------------+---------
↪------------+
```

(continues on next page)

(continued from previous page)

1.6.5 Querying the database with the CLI

ARA provides a CLI client to query the database.

Example commands:

```
$ ara help
usage: ara [--version] [-v | -q] [--log-file LOG_FILE] [-h] [--debug]

A CLI client to query ARA databases

optional arguments:
  --version            show program's version number and exit
  -v, --verbose        Increase verbosity of output. Can be repeated.
  -q, --quiet          Suppress output except warnings and errors.
  --log-file LOG_FILE  Specify a file to log output. Disabled by default.
  -h, --help           Show help message and exit.
  --debug              Show tracebacks on errors.

Commands:
  complete       print bash completion command
  data list      Returns a list of recorded key/value pairs
  data show      Show details of a recorded key/value pair
  file list      Returns a list of files
  file show      Show details of a file
  generate html  Generates a static tree of the web application
  generate junit Generate junit stream from ara data
  help           print detailed help for another command
  host facts     Show facts for a host
  host list      Returns a list of hosts
  host show      Show details of a host
  play list      Returns a list of plays
  play show      Show details of a play
  playbook delete  Delete playbooks from the database.
  playbook list  Returns a list of playbooks
  playbook show  Show details of a playbook
  result list    Returns a list of results
  result show    Show details of a result
  stats list     Returns a list of statistics
  stats show     Show details of a statistic
  task list      Returns a list of tasks
  task show      Show details of a task

# ara help result list
usage: ara result list [-h] [-f {csv,json,table,value,yaml}] [-c COLUMN]
                       [--max-width <integer>] [--noindent]
                       [--quote {all,minimal,none,nonnumeric}]

Returns a list of results

optional arguments:
  -h, --help            show this help message and exit

output formatters:
```

(continues on next page)

(continued from previous page)

```
  output formatter options

  -f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated

table formatter:
  --max-width <integer>
                        Maximum display width, 0 to disable

json formatter:
  --noindent            whether to disable indenting the JSON

CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric

# ara result list
+--------------------------------------+-----------+--------------------+---------+---
↪-----+---------+-------------+---------------+---------------------+----------------
↪-----+
| ID                                   | Host      | Task               | Changed |␣
↪Failed | Skipped | Unreachable | Ignore Errors | Time Start          | Time End     ␣
↪        |
+--------------------------------------+-----------+--------------------+---------+---
↪-----+---------+-------------+---------------+---------------------+----------------
↪-----+
| 79ee4b5b-667d-43a1-b10d-b48ebf422141 | localhost | Ping               | False   |␣
↪False  | False   | False       | False         | 2016-05-14 03:27:39 | 2016-05-14␣
↪03:27:39 |
| b3a04d9e-c9df-4126-8481-5bdb9d9795f7 | localhost | Really debug thing | False   |␣
↪False  | False   | False       | False         | 2016-05-14 03:27:39 | 2016-05-14␣
↪03:27:39 |
+--------------------------------------+-----------+--------------------+---------+---
↪-----+---------+-------------+---------------+---------------------+----------------
↪-----+

# ara result show b3a04d9e-c9df-4126-8481-5bdb9d9795f7 --long
+---------------+-------------------------------------------------------------+
| Field         | Value                                                       |
+---------------+-------------------------------------------------------------+
| ID            | b3a04d9e-c9df-4126-8481-5bdb9d9795f7                        |
| Host          | localhost                                                   |
| Task          | Really debug thing (1d24921e-bebc-4732-a362-32df24c8cb8b)   |
| Changed       | False                                                       |
| Failed        | False                                                       |
| Skipped       | False                                                       |
| Unreachable   | False                                                       |
| Ignore Errors | False                                                       |
| Time Start    | 2016-05-14 03:27:39                                         |
| Time End      | 2016-05-14 03:27:39                                         |
| Result        | {                                                           |
|               |     "_ansible_no_log": false,                               |
|               |     "_ansible_verbose_always": true,                        |
|               |     "changed": false,                                       |
|               |     "failed": false,                                        |
```

(continues on next page)

(continued from previous page)

```
|               |     "msg": "Really debug thing",                            |
|               |     "skipped": false,                                       |
|               |     "unreachable": false                                    |
|               | }                                                           |
+---------------+-------------------------------------------------------------+
```

1.6.6 Browsing the web interface

The web UI frontend is a visualization of the data recorded in the database. It provides insight on your playbooks, your hosts, your tasks and the results of your playbook run.

The interface provided by ARA provides is a simple Flask application. There are currently two documented options to host the web interface:

1. *Embedded development server* (easiest but least performance) 3. *Apache with mod_wsgi* (recommended)

These should be enough to get you started or help you choose your own path on other deployment options you might be used to when hosting Flask applications.

1.6.7 Generating a static HTML version of the web application

ARA is able to generate a static html version of it's dynamic, database-driven web application.

This can be useful if you need to browse the results of playbook runs without having to rely on the database backend configured.

For example, in the context of continuous integration, you could run an Ansible job with ARA, generate a static version and then recover the resulting build as artifacts of the jobs, allowing you to browse the results in-place.

This is done with the `ara generate html` command.

Note: Hosting statically generated reports is not very efficient at a large scale. Please refer to: *Advanced use cases*.

By default, ARA will generate a static version for all the recorded playbook runs in it's database. It is also possible to generate a report for one or many specific playbooks. This is done by retrieving the playbook IDs you are interested in with `ara playbook list` and then using the `ara generate html` command with the `--playbook` parameter:

```
$ ara help generate html
usage: ara generate html [-h] [--playbook <playbook> [<playbook> ...]] <path>

Generates a static tree of the web application

positional arguments:
  <path>                Path where the static files will be built in

optional arguments:
  -h, --help            show this help message and exit
  --playbook <playbook> [<playbook> ...]
                        Only include the specified playbooks in the
                        generation.

$ ara generate html /tmp/build/
Generating static files at /tmp/build/...
```

(continues on next page)

(continued from previous page)

```
Done.
$ tree /tmp/build/
/tmp/build/
├── host
│   ├── anotherhost
│   ├── index.html
│   └── localhost
├── index.html
├── play
│   └── play
│       └── 6ec9ef1d-dd73-4378-8347-1242f6be8f1e
├── playbook
│   ├── bf81a7db-b549-49d9-b10e-19918225ec60
│   │   ├── index.html
│   │   └── results
│   │       ├── anotherhost
│   │       │   ├── index.html
│   │       │   └── ok
│   │       └── localhost
│   │           ├── index.html
│   │           └── ok
│   └── index.html
├── result
│   ├── 136100f7-fba7-44ba-83fc-1194509ad2dd
│   ├── 37532523-b2ec-4931-bb73-3c7e5c6fa7bf
│   ├── 3cef2a10-8f41-4f01-bc49-12bed179d7e9
│   └── e3b7e172-c6e4-4ee4-b4bc-9a51ff84decb
├── static
│   ├── css
│   │   ├── ara.css
│   │   ├── bootstrap.min.css
│   │   └── bootstrap-theme.min.css
│   └── js
│       ├── bootstrap.min.js
│       └── jquery-2.2.3.min.js
└── task
    ├── 570fe763-69bb-4141-80d4-578189c5938b
    └── 946e1bc6-28b9-4f2f-ad4f-75b3c6c9032d

13 directories, 22 files
```

1.6.8 Generating a static junit version of the task results

ARA is able to generate a junit xml report that contains task results and their status.

This is done with the `ara generate junit` command.

By default, ARA will generate a report on all task results across all the recorded playbook runs in it's database. It is also possible to generate a report for one or many specific playbooks. This is done by retrieving the playbook IDs you are interested in with `ara playbook list` and then using the `ara generate junit` command with the `--playbook` parameter:

```
$ ara help generate junit
usage: ara generate junit [-h] [--playbook <playbook> [<playbook> ...]]
                          <output file>
```

(continues on next page)

(continued from previous page)

```
Generate junit stream from ara data

positional arguments:
  <output file>         The file to write the junit xml to. Use "-" for
                        stdout.

optional arguments:
  -h, --help            show this help message and exit
  --playbook <playbook> [<playbook> ...]
                        Only include the specified playbooks in the
                        generation.

$ ara generate junit -
<?xml version="1.0" ?>
<testsuites errors="0" failures="3" tests="66" time="33.0">
    <testsuite errors="0" failures="3" name="Ansible Tasks" skipped="5" tests="66"␣
↪time="33">
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="Deferred setup" time="3.000000"/>
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="include"/>
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="Ensure temporary directory exists"/>
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="Check if a file exists"/>
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="Touch a file if it doesn't exist"/>
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="Remove a file if it doesn't exist"/>
        <testcase classname="localhost._home_dev_ara_ara_tests_integration_smoke_yml.
↪ARA_Tasks_test_play" name="Remove a file if it exists">
[...]
```

1.6.9 Generating a static subunit version of the task results

ARA is able to generate a subunit report that contains task results and their status.

This is done with the `ara generate subunit` command.

By default, ARA will generate a report on all task results across all the recorded playbook runs in it's database. It is also possible to generate a report for one or many specific playbooks. This is done by retrieving the playbook IDs you are interested in with `ara playbook list` and then using the `ara generate subunit` command with the `--playbook` parameter:

```
$ ara help generate subunit
usage: ara generate subunit [-h] [--playbook <playbook> [<playbook> ...]]
                            <output file>

Generate subunit binary stream from ARA data

positional arguments:
  <output file>         The file to write the subunit binary stream to. Use
                        "-" for stdout.

```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  --playbook <playbook> [<playbook> ...]
                        Only include the specified playbooks in the
                        generation.

$ ara generate subunit - | subunit2csv
test,status,start_time,stop_time
50d4e04fe034bea7479bc4a3fa3703254298baa8,success,2017-07-28 03:07:21+00:00,2017-07-28␣
↪03:07:21+00:00
a62f7a36683972efe1ef6e51e389417521502153,success,2017-07-28 03:07:22+00:00,2017-07-28␣
↪03:07:22+00:00
8902778f958439806aee2a22c26d8b79dc61c964,success,2017-07-28 03:07:22+00:00,2017-07-28␣
↪03:07:22+00:00
fd2d199b22b635ed82b41d5edf8c1774f64484dc,success,2017-07-28 03:07:22+00:00,2017-07-28␣
↪03:07:22+00:00
[...]
```

1.7 Contributing

ARA is an Open Source project and welcomes contributions, whether they are in the form of feedback, comments, suggestions, bugs, code contributions or code reviews.

ARA does not use GitHub for issues or pull requests.

The project has decided to be hosted under the OpenStack umbrella to benefit from the code review and testing infrastructure on which hundreds of developers contribute to hundreds of projects every day.

This proven infrastructure brings with it a robust contribution workflow to be able to contribute, review, test and merge code easily and efficiently.

The end result is higher standards, better code, more testing, less regressions and more stability.

If you are familiar with the process of contributing to an OpenStack project, ARA is no different. If this is something new for you, you should be excited and read on.

This documentation you will find here is mostly a summary of OpenStack's developer getting started guide.

Note: ARA is *not* an official OpenStack project. As such, you are not required to have signed a contributors agreement with the OpenStack foundation to be able to contribute to ARA.

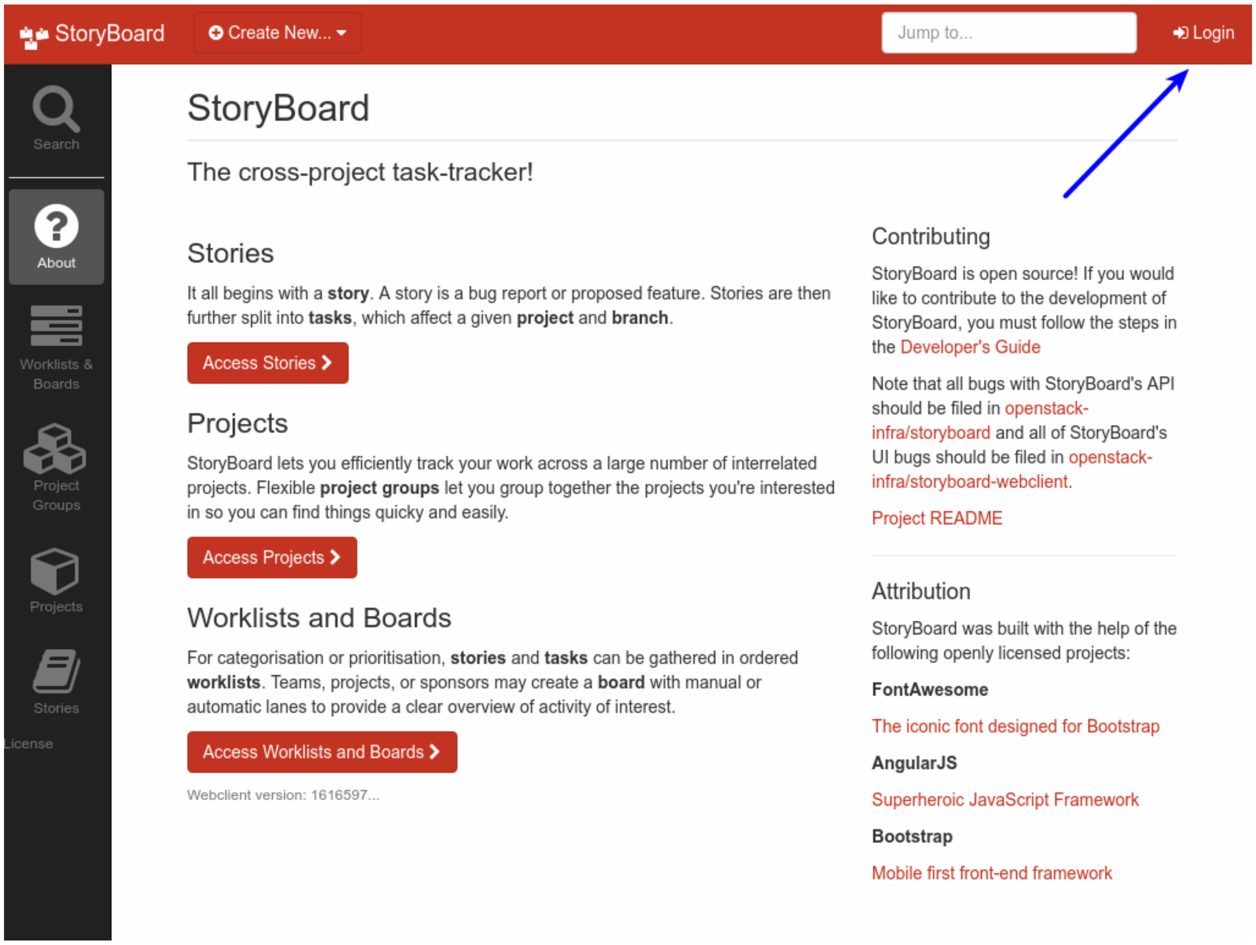
1.7.1 Set up your Ubuntu Launchpad account

OpenStack's Gerrit and StoryBoard instances currently use Launchpad for authentication. If you do not already have a Launchpad account, you will need to create one here.

1.7.2 Filing issues and bugs

Once you have your Ubuntu Launchpad account set up, you're ready to start contributing to the ARA project tracker in StoryBoard.

First, you'll need to login to StoryBoard – the ARA project can be found here: https://storyboard.openstack.org/#!/project/843



StoryBoard

The cross-project task-tracker!

Stories

It all begins with a **story**. A story is a bug report or proposed feature. Stories are then further split into **tasks**, which affect a given **project** and **branch**.

Projects

StoryBoard lets you efficiently track your work across a large number of interrelated projects. Flexible **project groups** let you group together the projects you're interested in so you can find things quicky and easily.

Worklists and Boards

For categorisation or prioritisation, **stories** and **tasks** can be gathered in ordered **worklists**. Teams, projects, or sponsors may create a **board** with manual or automatic lanes to provide a clear overview of activity of interest.

Webclient version: 1616597...

Contributing

StoryBoard is open source! If you would like to contribute to the development of StoryBoard, you must follow the steps in the Developer's Guide

Note that all bugs with StoryBoard's API should be filed in openstack-infra/storyboard and all of StoryBoard's UI bugs should be filed in openstack-infra/storyboard-webclient.

Project README

Attribution

StoryBoard was built with the help of the following openly licensed projects:

FontAwesome

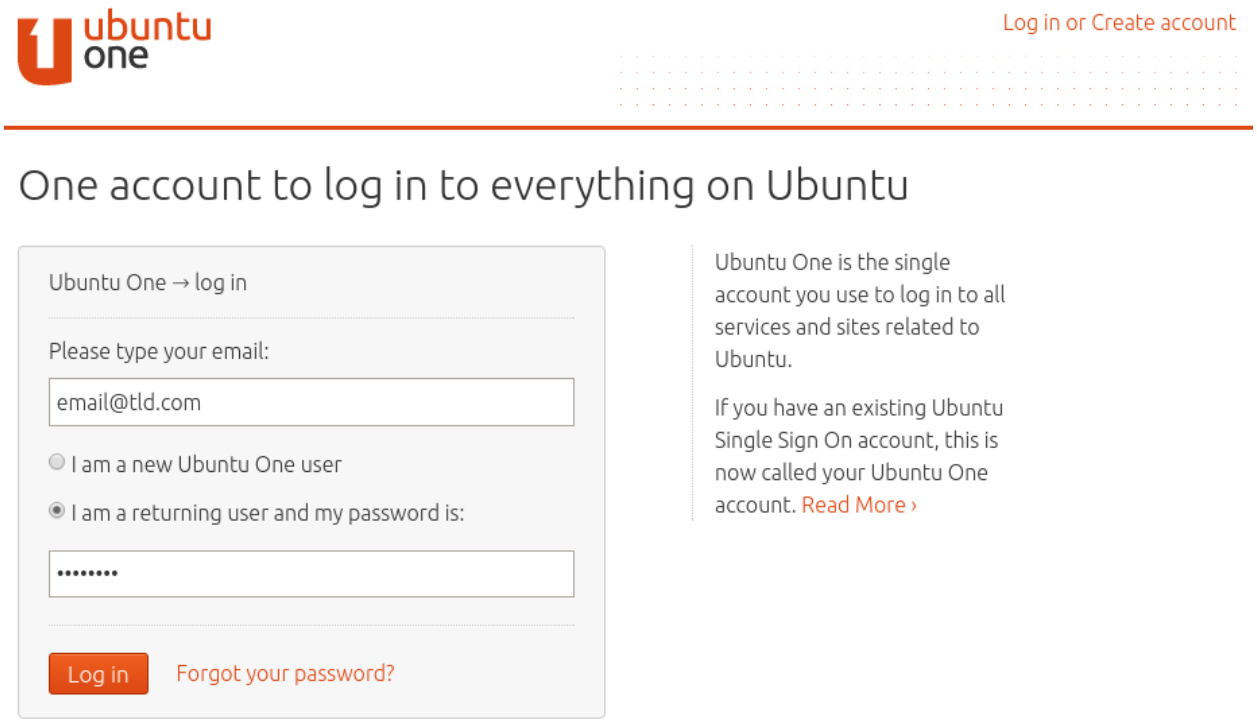
The iconic font designed for Bootstrap

AngularJS

Superheroic JavaScript Framework

Bootstrap

Mobile first front-end framework



One account to log in to everything on Ubuntu

Ubuntu One → log in

Please type your email:

email@tld.com

I am a new Ubuntu One user

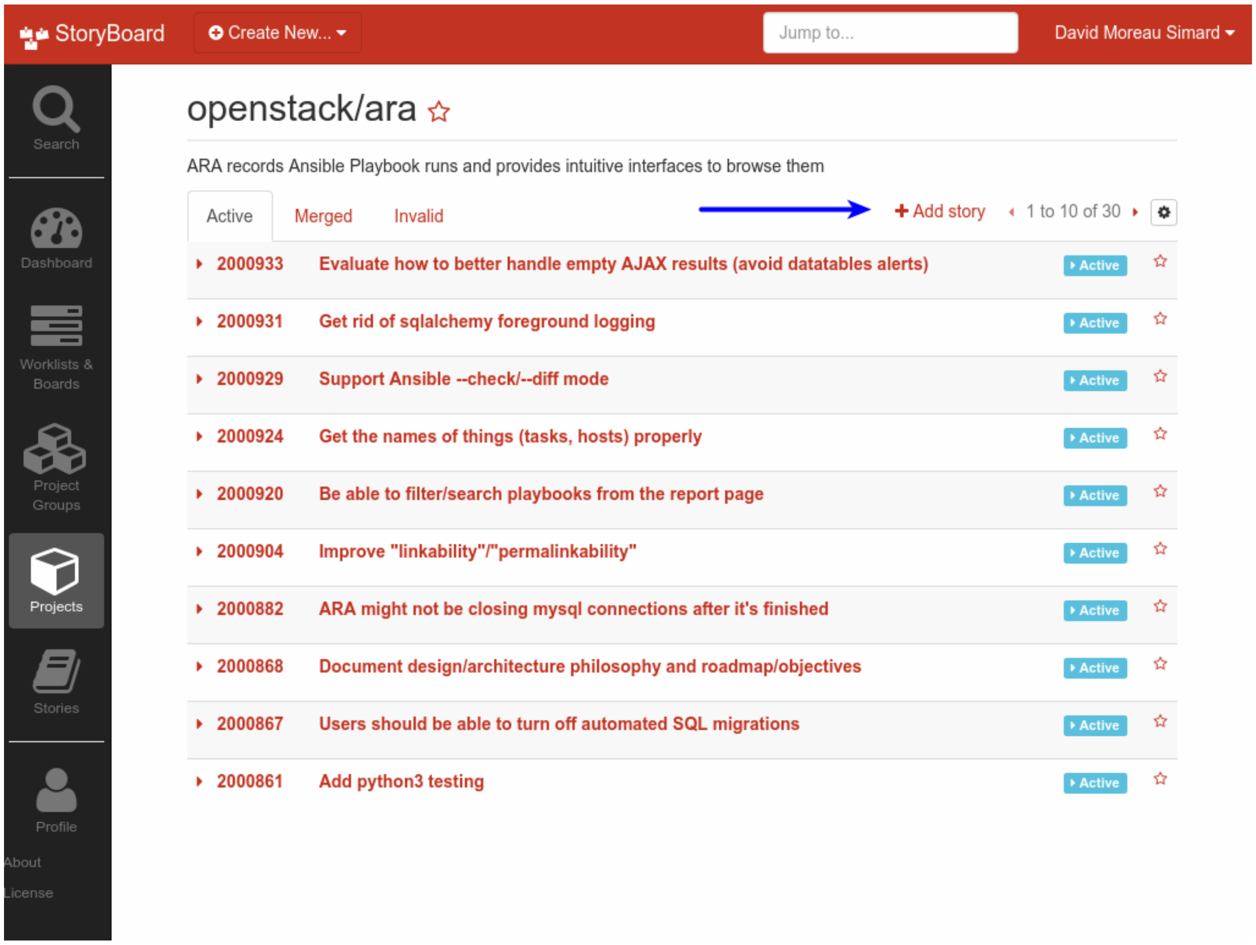
I am a returning user and my password is:

••••••••

Ubuntu One is the single account you use to log in to all services and sites related to Ubuntu.

If you have an existing Ubuntu Single Sign On account, this is now called your Ubuntu One account. Read More ›

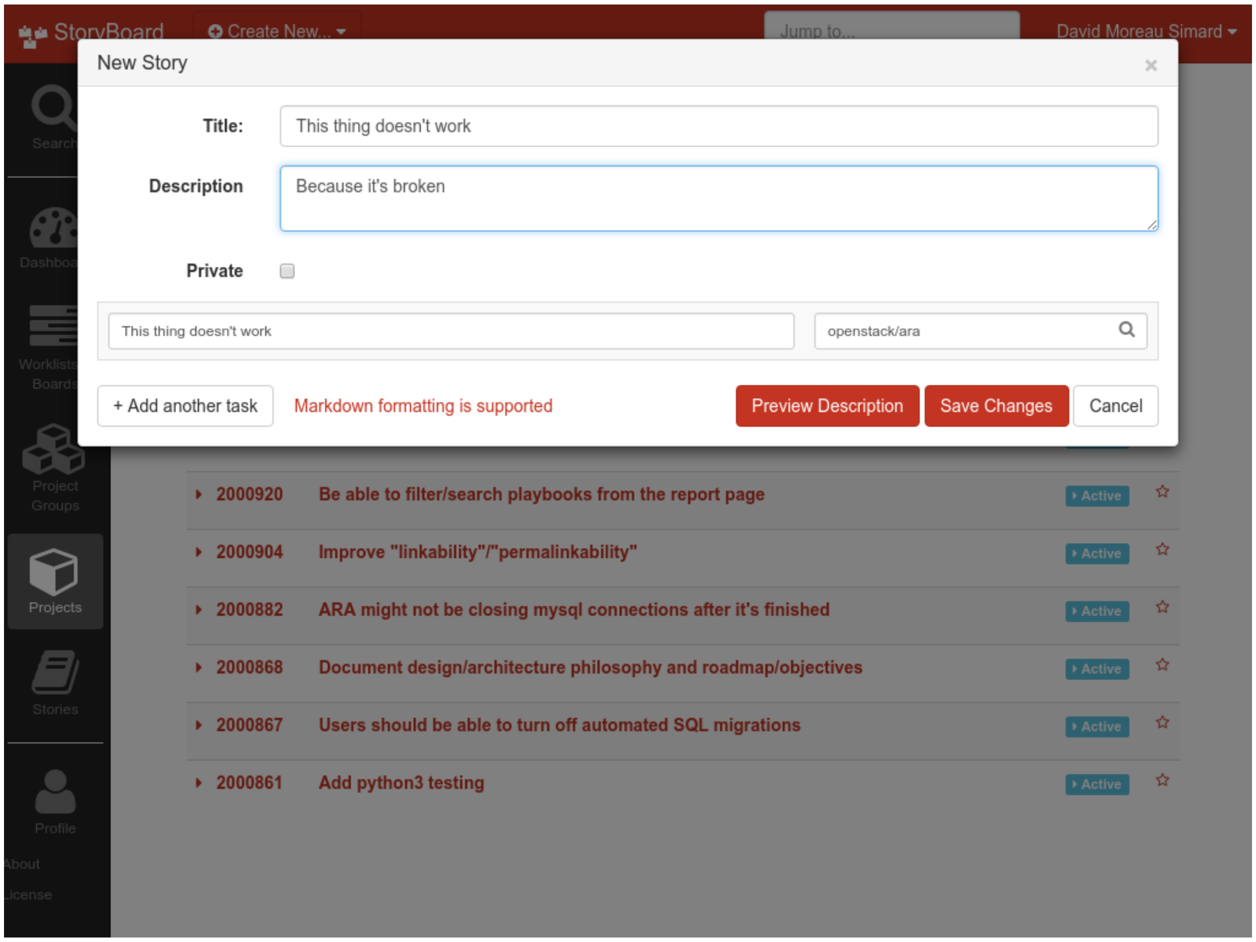
Once you're logged in, you'll want to create a story for the `openstack/ara` project:



openstack/ara

ARA records Ansible Playbook runs and provides intuitive interfaces to browse them

2000933	Evaluate how to better handle empty AJAX results (avoid datatables alerts)	Active
2000931	Get rid of sqlalchemy foreground logging	Active
2000929	Support Ansible --check/--diff mode	Active
2000924	Get the names of things (tasks, hosts) properly	Active
2000920	Be able to filter/search playbooks from the report page	Active
2000904	Improve "linkability"/"permalinkability"	Active
2000882	ARA might not be closing mysql connections after it's finished	Active
2000868	Document design/architecture philosophy and roadmap/objectives	Active
2000867	Users should be able to turn off automated SQL migrations	Active
2000861	Add python3 testing	Active



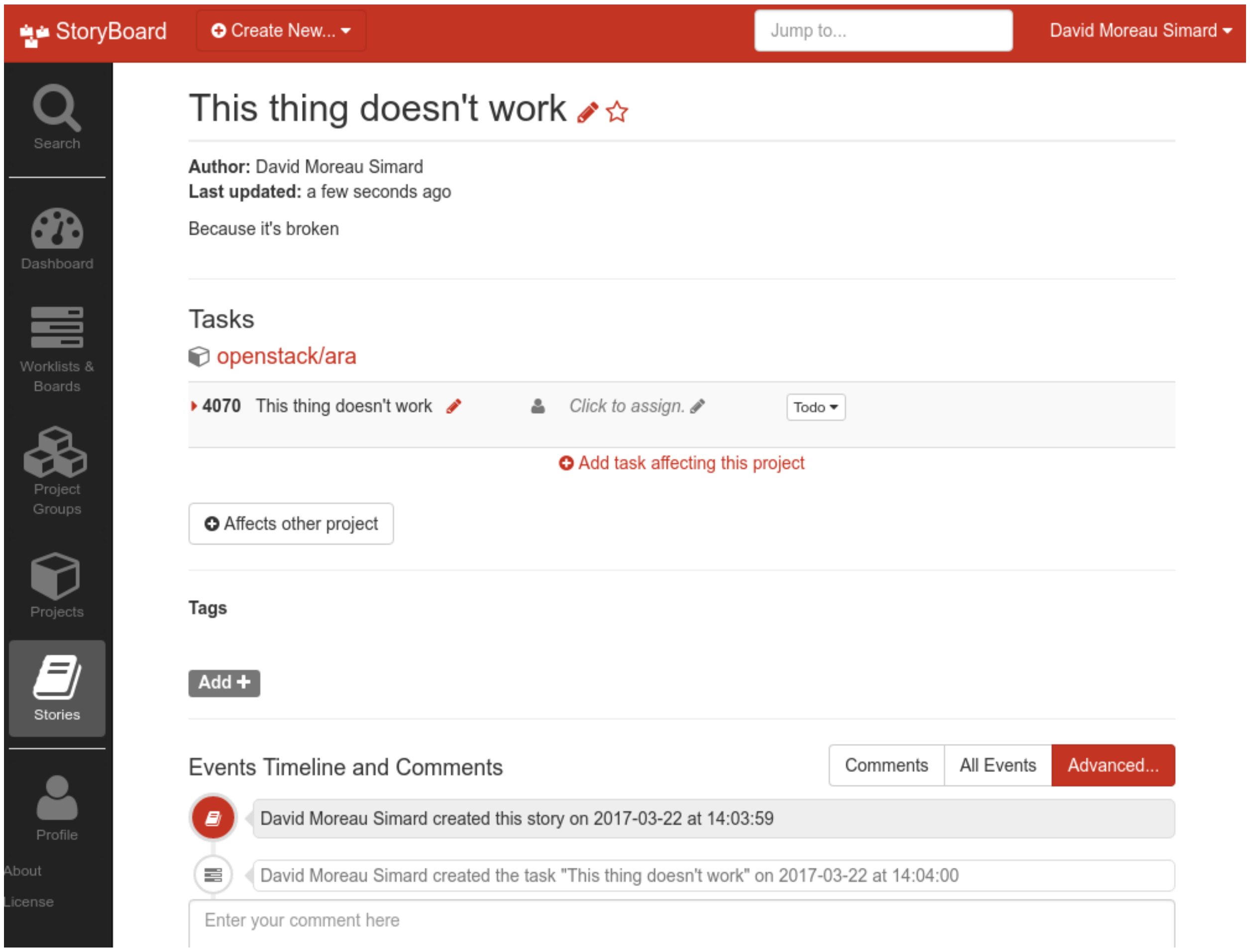
Title: This thing doesn't work

Description: Because it's broken

This thing doesn't work | openstack/ara

2000920	Be able to filter/search playbooks from the report page	Active
2000904	Improve "linkability"/"permalinkability"	Active
2000882	ARA might not be closing mysql connections after it's finished	Active
2000868	Document design/architecture philosophy and roadmap/objectives	Active
2000867	Users should be able to turn off automated SQL migrations	Active
2000861	Add python3 testing	Active

And then you're done:



This thing doesn't work

Author: David Moreau Simard
Last updated: a few seconds ago

Because it's broken

Tasks

openstack/ara

4070 This thing doesn't work — Click to assign. — Todo

Add task affecting this project

Affects other project

Tags

Add +

Events Timeline and Comments — Comments | All Events | Advanced...

David Moreau Simard created this story on 2017-03-22 at 14:03:59

David Moreau Simard created the task "This thing doesn't work" on 2017-03-22 at 14:04:00

Enter your comment here

1.7.3 Contributing code or code reviews

Set up your Gerrit code review account

If you'll be contributing code or code reviews, you'll need to set up your Gerrit code review account.

Once you have your Launchpad account, you will be able to sign in to review.openstack.org.

To be able to submit code, Gerrit needs to have your public SSH key in the same way Github does. To do that, click on your name at the top right and go to the settings where you will see the tab to set up your SSH key.

Installing Git Review

Git Review is a python module that adds a "git review" command that wraps around the process of sending a commit for review in Gerrit. You need to install it to be able to send patches for code reviews.

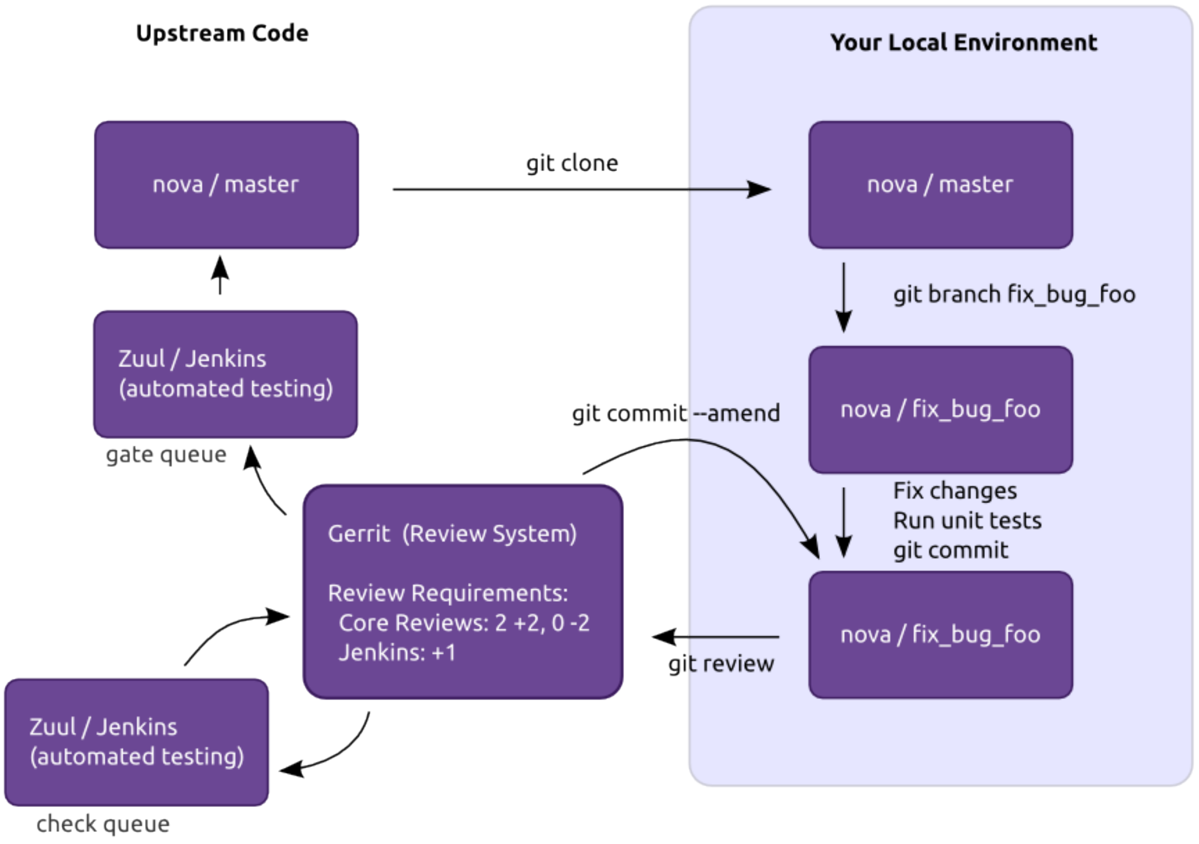
There are different ways to install git-review, choose your favorite.

Install development dependencies

ARA requires some additional dependencies for development purposes, for running tests, for example.

Make sure they are installed according to the *documentation*.

1.7.4 Sending a patch for review



The process looks a bit like this:

```
$ git clone https://github.com/openstack/ara
$ cd ara
$ git checkout -b super_cool_feature

# << hack on super_cool_feature >>

$ git commit -a --message="This is my super cool feature"
$ git review
```

When you send a commit for review, it'll create a code review request in Gerrit for you. When that review is created, it will automatically be tested by a variety of jobs that the ARA maintainers have set up to test every patch that is sent.

We'll check for things like code quality (pep8/flake8), run unit tests to catch regressions and we'll also run both integration tests on different operating systems to make sure everything really works.

The result of the tests are added as a comment in the review when all of them are completed. If you're interested in digging into the logs for a particular test, clicking on the results of the test will take you to console, debug logs and a built version of ARA's web interface.

If you get a failed test result and you believe you have fixed the issue, add the files, amend your commit (`git commit --amend`) and send it for review once again. This will create a new patchset that will be up for review and testing.

To be able to merge a patch, the tests have to come back successful and the core reviewers must provide their agreement with the patch.

1.7.5 Running tests locally

Unit tests:

```
# Python 2.7
tox -e py27
# Python 3.5
tox -e py35
```

pep8/flake8/bandit/bashate tests:

```
tox -e pep8
```

Documentation tests:

```
# This will also build the docs locally in docs/build/html
tox -e docs
```

Integration tests:

At the root of the ARA source, you'll find the `run_tests.sh` script that allows you to easily run integration tests across a range of different configurations.

ARA's integration tests do not require superuser privileges, are all self-contained in temporary directories and python virtual environments. They are designed to safely and easily run either on your local machine or in a CI environment such as Jenkins.

Here's how you would `run_tests.sh` to run integration tests:

```
$ ./run_tests.sh -h
usage: ./run-tests.sh [-a|--ansible ANSIBLE_VERSION] [-a|--python PYTHON_VERSION] [-
↪h|--help]

Runs ARA integration tests

optional arguments:
-a, --ansible      Ansible version to test with (ex: '2.3.1.0', 'devel')
                   Defaults to version in requirements.txt (latest version of Ansible)
-p, --python       Python version from a tox environment to test with (ex: 'py27',
↪'py35')
                   Defaults to py27
-h, --help         Prints this help dialog.

# With the default configuration (latest release of Ansible and py27)
$ ./run_tests.sh
# or.. with the devel version of Ansible with py35
$ ./run_tests.sh -a devel -p py35
```

PostgreSQL integration tests:

In order to get `run_tests.sh` to run PostgreSQL integration tests, you'll need to set a few environment variables:

```
export ARA_TEST_PGSQL=1
export ARA_TEST_PGSQL_USER=ara
export ARA_TEST_PGSQL_PASSWORD=password
```

You'll also need development headers for PostgreSQL to build psycopg2, the defacto pgsql adapter for Python.

To install the package on Ubuntu/Debian:

```
sudo apt install postgresql-server-dev-9.5
```

To install the package on RHEL/CentOS/Fedora:

```
sudo yum install postgresql-devel
```

If you need an ephemeral PostgreSQL server to test against, you can spin one up with Docker easily:

```
docker run --name ara_pgsql \
     -e POSTGRES_USER=${ARA_TEST_PGSQL_USER} \
     -e POSTGRES_PASSWORD=${ARA_TEST_PGSQL_PASSWORD} \
     -e POSTGRES_DB=ara \
     -p 5432:5432 \
     -d postgres:alpine
```

Once the PostgreSQL is up and the environment variables are set, you're ready to run integration tests as usual:

```
./run_tests.sh
```

1.7.6 More reading

- Official OpenStack developer documentation
- Gerrit documentation
- Git commit good practices

1.8 Manifesto: Project core values

ARA is an open source project that was created by Ansible users for Ansible users.

Its purpose is to provide a way to simply and easily understand what happens throughout playbook runs at any scale.

ARA itself is composed of several components to achieve that purpose. The project as well as those components adhere to some important core values.

This manifesto exists to explain the different core values incorporated in the project's development and roadmap for users, contributors and developers alike.

1.8.1 1) Simplicity is fundamental

In the Zen of Python, you'll find the following:

> **Simple is better than complex**

This is paramount to the project. ARA should always be simple to install, simple to use and simple to understand.

Simplicity is also expressed in terms of configurability: ARA should come with sane and working defaults out of the box.

It should be simple (but not required) to customize the behavior of ARA. This is why ARA can be configured using the exact same means as Ansible.

1.8.2 2) Do one thing and do it well

The scope of the ARA project is narrow on purpose and is strongly aligned with one of the values from the UNIX philosophy:

> **Write programs that do one thing and do it well**

ARA records Ansible playbook runs and makes the recorded data available and intuitive for users and systems.

A narrow project scope for ARA allows developers and users to focus on a limited feature set in order to ensure each component is built and usable both simply and optimally.

1.8.3 3) Empower users to get their work done

This core value of the project is about being receptive to user feedback and understanding what they need.

ARA should provide generic implementations to allow them to get their work done while keeping in mind the two previous core values.

This warrants examples in order to have a common understanding of what this means:

- ARA does not provide additional data beyond what is sent and made available by Ansible directly. Ansible upstream modules can be improved to send more information that would then be made available to ARA.
- ARA does not "connect" directly to systems such as Logstash but provides machine-readable output through its command line interface (CLI), allowing users to feed data easily to the system of their choosing.
- ARA does not tell you which host Ansible ran from or automatically discover the git versions of your playbooks but allows you to save arbitrary data in its database for future reference.

1.8.4 4) Don't require users to change their workflows

ARA should never require users to change how they already use Ansible beyond installing and configuring Ansible to use ARA.

ARA should be a drop-in, seamless and transparent addition to their workflows.

1.8.5 5) De-centralized, offline and standalone by default

It should never be required to run Ansible with ARA from one single, unique and central location.

Users should be able to record data no matter where Ansible runs, whether it is on their laptops, workstations, servers, virtual machines, etc.

ARA should provide the means to easily aggregate collected data in the form of a centralized relational database but it should default to a standalone, offline and self-contained mode of operation.