TosKer Documentation

Release 2.0.2

luca

Apr 06, 2018

Contents


TosKer

CHAPTER 1

TosKer

TosKer is an orchestrator engine capable of automatically deploying and managing multi-component applications specifies in OASIS TOSCA, by exploiting Docker as a lightweight virtualization framework. The novelty of TosKer is to decouple the application-specific components, from the containers used to build their infrastructure. This permits to improve the orchestration of the components and to ease the change of the containers underneath.

Contents:

1.1 Installation

1.1.1 Stable release

To install TosKer, run this command in your terminal:

```
$ pip install tosker
```

This is the preferred method to install new TosKer, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.1.2 From sources

The sources for TosKer can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone https://github.com/di-unipi-socc/TosKer
```

Or download the tarball:

```
$ curl  -OL https://github.com/di-unipi-socc/TosKer/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

1.2 Usage

1.2.1 Complete Usage

This is the complete usage of TosKer:

Usage: tosker [OPTIONS] COMMAND [ARGS]...

Orchestrate TOSCA applications on top of Docker.

TosKer is an orchestrator engine capable of automatically deploying and managing multi-component applications specified in OASIS TOSCA. The engine executes the component exploiting Docker as a lightweight virtualization framework.

Options:

--version	Show the version and exit.
-q, --quiet	Prevent the command to print.
--debug	Print additional debuging log (override quiet mode).
--help	Show this message and exit.

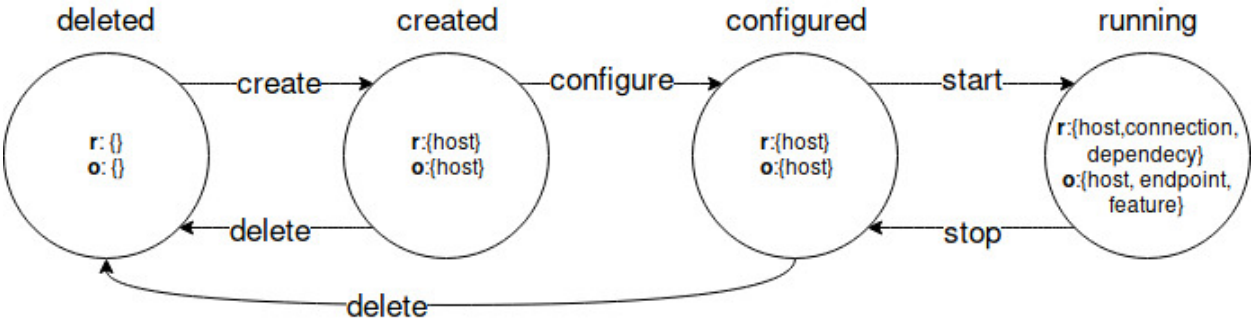
Commands: exec Exec a plan. log Print the execution log of an operation. ls List all the deployed applications. prune Remove all files, container and volumes...

1.3 Management Protocols

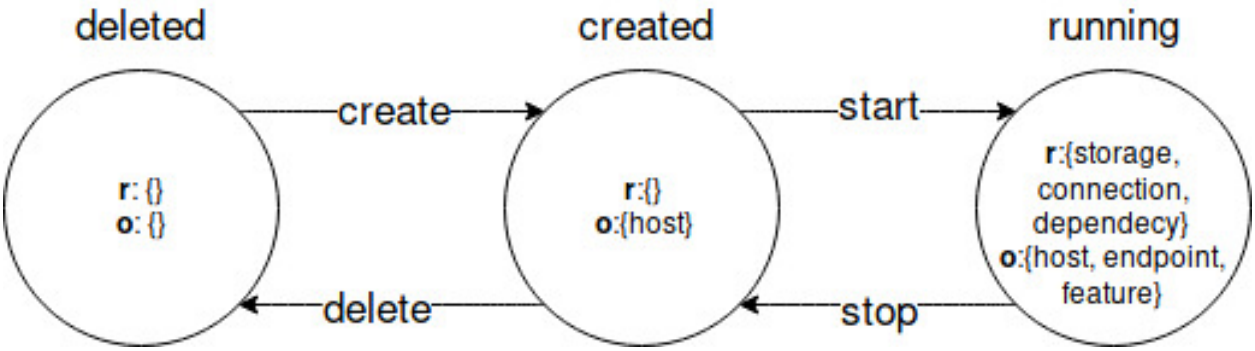
TosKer implement the management protocols to orchestrate the component of the application.

1.3.1 Default Protocols

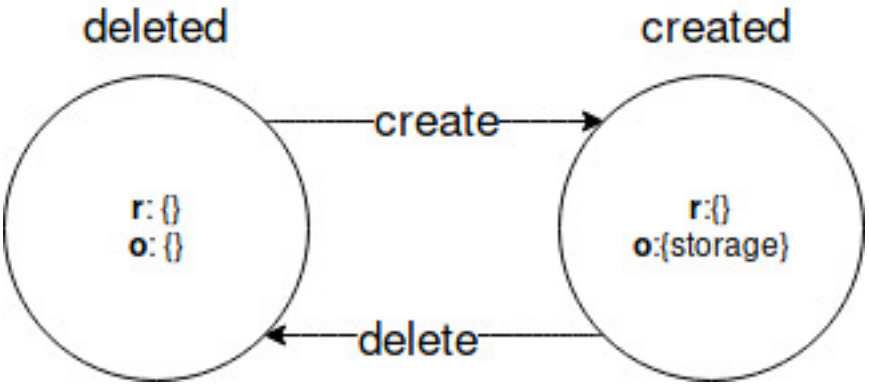
Software



Container



Volume



1.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

1.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/di-unipi-socc/TosKer/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

new TosKer could always use more documentation, whether as part of the official new TosKer docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/di-unipi-socc/TosKer/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

1.4.2 Get Started!

Ready to contribute? Here's how to set up *TosKer* for local development.

1. Fork the *TosKer* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/TosKer
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv tosker
$ cd TosKer/
$ pip install -e '.[dev]'
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 tosker tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

1.4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in DESCRIPTION.rst.
3. The pull request should work for Python 2.7, 3.3, 3.4, 3.5, 3.6 and for PyPy. Check https://travis-ci.org/lucarin91/TosKer/pull_requests and make sure that the tests pass for all supported Python versions.

1.4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_tosker
```

1.5 Credits

1.5.1 Development Lead

- lucarin91 <to@lucar.in>

1.5.2 Contributors

None yet. Why not be the first?

1.6 History

1.6.1 2.0.2 (2018-02-12)

Stable release with Management Protocols.

- Add support of two type of plans (.plan, .csv).
- Fix piped input error.
- Fix errors in python2 interpreter.
- Fix bug that does not execute the delete operation on Docker volumes.

1.6.2 2.0.1 (2017-12-09)

- Switch to Management Protocols to manage the life cycle of the components
- Add support for derived node types.
- Add support for custom interfaces.
- Support custom management protocol defined using policies.
- Support safe execution of plans (list of <component, interface, operation>).
- Improve command line interface.

1.6.3 1.0.1 (2017-11-20)

Stable release without Management Protocols.

- Add command log, to show the execution of an operation on a component.
- Add command prune, to remove all TosKer files and restore initial state.
- Improve memory management.
- Improve command line interface.
- Bug fix.

1.6.4 0.4.0 (2017-07-10)

- First release on PyPI.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search