# supervisorclusterctl Documentation

***Release 0.1.1***

**Robert Winkler**

April 22, 2015

Contents


**1 Installation** **3**
- 1.1 Control Machine Requirements . . . 3
- 1.2 Managed Node Requirements . . . 3
- 1.3 Installing supervisorclusterctl . . . 4

**2 User Guide** **5**
- 2.1 Running supervisorclusterctl . . . 5

**3 Contributing** **7**
- 3.1 Types of Contributions . . . 7
- 3.2 Get Started! . . . 8
- 3.3 Pull Request Guidelines . . . 8

**4 Authors** **9**
- 4.1 Development Lead . . . 9
- 4.2 Contributors . . . 9

**5 Changes** **11**
- 5.1 0.1.2 . . . 11
- 5.2 0.1.1 . . . 11
- 5.3 0.1.0 . . . 11

supervisorclusterctl is a cmd line tool that allows to control a cluster of processes by utilizing Supervisor and Ansible.

The tool uses Ansible and Supervisor's supervisorctl to control remote Supervisor daemons (supervisord). The primary goal of supervisorclusterctl is to simplify the use of Ansible and Supervisor's supervisorctl by focusing on the most commonly used supervisorctl actions. The tool only works with Python 2.6 or 2.7, since Ansible does not work with Python 3 yet.

CHAPTER 1

# Installation

**Topics**


- Installation
  - Control Machine Requirements
  - Managed Node Requirements
  - Installing supervisorclusterctl
    - Installing via pip
    - Installing from source
    - Create RPM package
    - Installing from tarball or zipball


## 1.1 Control Machine Requirements

supervisorclusterctl requires Ansible to be installed on the control machine. You only need to install Ansible on one machine and it can control an entire fleet of remote nodes from that central point.

Currently Ansible can be run from any machine with Python 2.6 installed (Windows isn't supported for the control machine). This includes Red Hat, Debian, CentOS, OS X, any of the BSDs, and so on.

Ansible can be installed via "pip", the Python package manager. If 'pip' isn't already available in your version of Python, you can get pip by:

```
$ sudo easy_install pip
```

Then install Ansible with:

```
$ sudo pip install ansible
```

**Note:** Some Linux distributions offer a version of Ansible that is installable through the system package manager. Use the package management tools of your distribution to check availability.

## 1.2 Managed Node Requirements

supervisorclusterctl requires Supervisor to be installed on the remote nodes.

Supervisor can be installed via "pip":

```
$ sudo pip install supervisor
```

**Note:** Some Linux distributions offer a version of Supervisor that is installable through the system package manager. Use the package management tools of your distribution to check availability.

## 1.3 Installing supervisorclusterctl

### 1.3.1 Installing via pip

supervisorclusterctl can be installed via "pip":

```
$ sudo pip install supervisorclusterctl
```

### 1.3.2 Installing from source

supervisorclusterctl is trivially easy to install from source. No daemons or database setup are required.

To install from source:

```
$ git clone https://github.com/RobWin/supervisorclusterctl.git
$ cd ./supervisorclusterctl
$ sudo python setup.py install
```

### 1.3.3 Create RPM package

You can create a RPM package which can be used by many of popular Linux distributions, including Red Hat, SuSE:

```
$ sudo python setup.py bdist_rpm
```

### 1.3.4 Installing from tarball or zipball

A zipball or tarball of the source are available on the Project page.

CHAPTER 2

# User Guide

**Topics**


- User Guide
  - Running supervisorclusterctl


## 2.1 Running supervisorclusterctl

Now that you've installed supervisorclusterctl, it's time to get started with some basics.

Run help to see all available command-line commands, arguments and options

```
$ supervisorclusterctl -h
     usage: supervisorclusterctl [-h] [-v] [-s] [-V]
                                 host-pattern
                                 {status,reread,reload,update,start,stop,restart,remove}
                                 ...

     supervisorclusterctl is a cmd line tool that allows to control a cluster of
     processes by utilizing Supervisor and Ansible.

     positional arguments:
       host-pattern          A host-pattern usually refers to a group of hosts. For
                             more details, see Ansible documentation about
                             Patterns.
       {status,reread,reload,update,start,stop,restart,remove}
                             One of the available supervisorctl actions.
         status              Get status info of all processes.
         reread              Reread the configuration files of supervisord
         reload              Restart remote supervisord
         update              Reload the configuration files of supervisord and
                             add/remove processes as necessary
         start               Start a process by name
         stop                Stop a process by name
         restart             Restart a process by name
         remove              Remove a process by name

     optional arguments:
       -h, --help            show this help message and exit
       -v, --verbose         run in verbose mode (-vvv for more, -vvvv to enable
```

```
                          connection debugging)
  -s, --sudo              run supervisorctl actions with sudo (nopasswd))
  -V, --version           show program's version number and exit
```

Run subcommand help to see all available arguments and options of the subcommand

```
$ supervisorclusterctl production restart -h
    usage: supervisorclusterctl host-pattern start [-h] process-name

    positional arguments:
      process-name  Name of the process

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

Now edit /etc/ansible/hosts and put one or more remote systems in it, for which you have your SSH key in `authorized_keys`:

```
[production]
192.168.0.11
192.168.0.12
[development]
192.168.0.10
```

This is an inventory file, which is explained in greater detail in Ansible's documentation

Run the `supervisorclusterctl <host-pattern> status` command to get the status of all Supervisor managed processes on all of your configured production nodes:

```
$ supervisorclusterctl production status
    192.168.0.11 | success | rc=0 >>
    managed_service         RUNNING    pid 12136, uptime 21:50:02

    192.168.0.12 | success | rc=0 >>
    managed_service         RUNNING    pid 12261, uptime 21:50:02
```

Now you are able to restart the `managed_service` process on all of your configured production nodes:

```
$ supervisorclusterctl production restart managed_service
    192.168.0.11 | success | rc=0 >>
    managed_service: stopped
    managed_service: started

    192.168.0.12 | success | rc=0 >>
    managed_service: stopped
    managed_service: started
```

CHAPTER 3

# 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:

## 3.1 Types of Contributions

### 3.1.1 Report Bugs

Report bugs at https://github.com/RobWin/supervisorclusterctl/issues.

If you are reporting a bug, please include:

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

### 3.1.2 Fix Bugs

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

### 3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

### 3.1.4 Write Documentation

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

### 3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/RobWin/supervisorclusterctl/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 :)

## 3.2 Get Started!

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

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

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

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 supervisorclusterctl
   $ cd supervisorclusterctl/
   $ python setup.py develop
   ```

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 are working:

   ```
   $ python setup.py test
   ```

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.

## 3.3 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.
3. The pull request should work for Python 2.6 and 2.7. Check https://travis-ci.org/RobWin/supervisorclusterctl/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 4

# Authors

## 4.1 Development Lead

- Robert Winkler <rbrtwnklr@gmail.com>

## 4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

# Changes

## 5.1 0.1.2

- Fixed issue #1: reload not working

## 5.2 0.1.1

- supervisorclusterctl returns the return code of the ansible call now
- Added new unit tests and fixed the setup.py

## 5.3 0.1.0

- Initial release.