

---

# **Charm Helpers Documentation**

*Release 0.19.13*

**Charm Helpers Developers**

**Apr 24, 2019**



---

# Contents

---

<b>1</b>	<b>Getting Started</b>	<b>3</b>
1.1	Installing Charm Tools . . . . .	3
1.2	Creating a New Charm . . . . .	3
1.3	Updating Charmhelpers Packages . . . . .	4
1.4	Next Steps . . . . .	6
<b>2</b>	<b>Examples</b>	<b>7</b>
2.1	Interacting with Charm Configuration . . . . .	7
2.2	Managing Charms with the Services Framework . . . . .	9
<b>3</b>	<b>API Documentation</b>	<b>13</b>
3.1	charmhelpers.core package . . . . .	13
3.2	charmhelpers.contrib package . . . . .	42
3.3	charmhelpers.fetch package . . . . .	86
3.4	charmhelpers.payload package . . . . .	89
3.5	charmhelpers.cli package . . . . .	90
3.6	charmhelpers.coordinator package . . . . .	91
<b>4</b>	<b>Contributing</b>	<b>97</b>
4.1	Source . . . . .	97
4.2	Submitting a Merge Proposal . . . . .	97
4.3	Open Bugs . . . . .	98
4.4	Documentation . . . . .	98
4.5	Getting Help . . . . .	98
<b>5</b>	<b>Changelog</b>	<b>99</b>
5.1	0.19.13 . . . . .	99
5.2	0.19.12 . . . . .	99
5.3	0.19.11 . . . . .	99
5.4	0.19.10 . . . . .	100
5.5	0.19.9 . . . . .	100
5.6	0.19.8 . . . . .	100
5.7	0.19.7 . . . . .	100
5.8	0.19.6 . . . . .	101
5.9	0.19.5 . . . . .	101
5.10	0.19.4 . . . . .	101
5.11	0.19.3 . . . . .	102

5.12	0.19.2	102
5.13	0.19.1	103
5.14	0.19.0	103
5.15	0.18.11	103
5.16	0.18.9	103
5.17	0.18.8	104
5.18	0.18.7	104
5.19	0.18.6	105
5.20	0.18.5	105
5.21	0.18.4	105
<b>6</b>	<b>Indices and tables</b>	<b>107</b>
	<b>Python Module Index</b>	<b>109</b>

The `charmhelpers` Python library is an extensive collection of functions and classes for simplifying the development of [Juju Charms](#). It includes utilities for:

- Interacting with the host environment
- Managing hook events
- Reading and writing charm configuration
- Installing dependencies
- Much, much more!



For a video introduction to `charmhelpers`, check out this [Charm School session](#). To start using `charmhelpers`, proceed with the instructions on the remainder of this page.

### 1.1 Installing Charm Tools

First, follow [these instructions](#) to install the `charm-tools` package for your platform.

### 1.2 Creating a New Charm

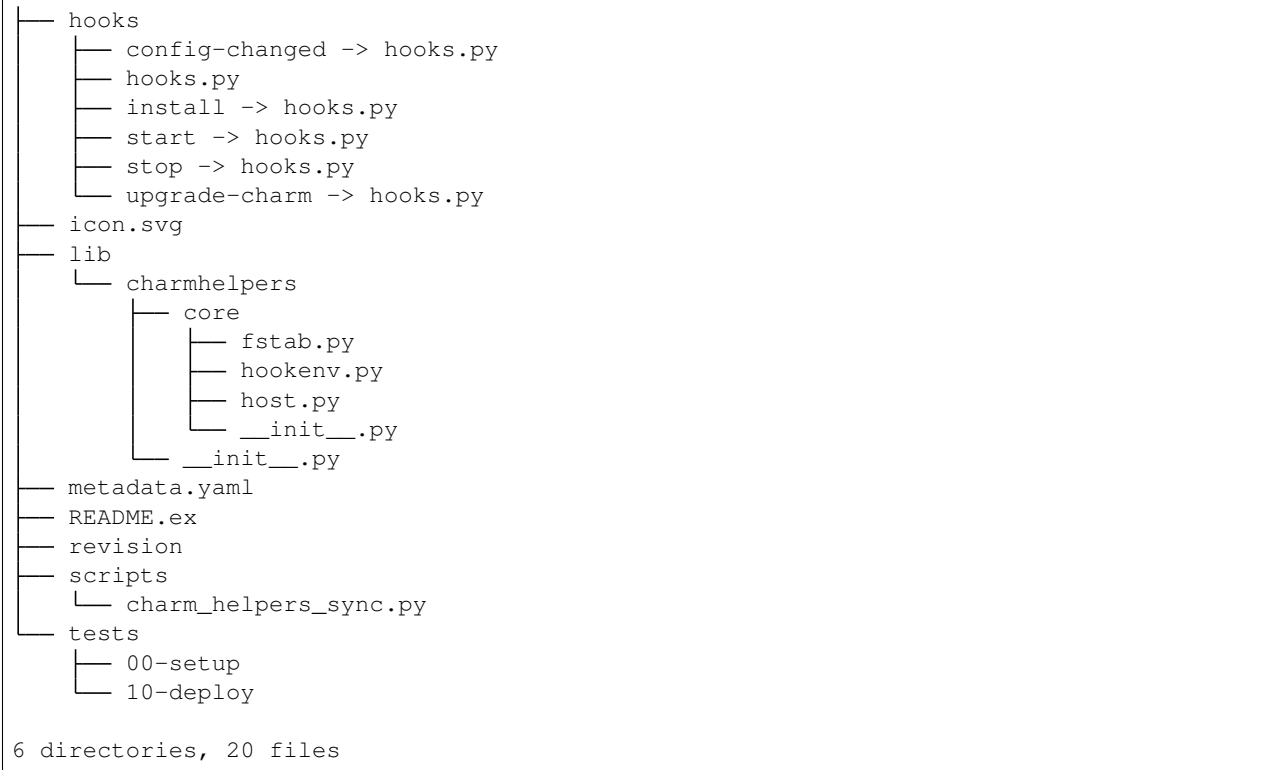
```
$ cd ~
$ mkdirs -p charms/precise
$ cd charms/precise
$ charm create -t python mycharm
INFO: Generating template for mycharm in ./mycharm
INFO: No mycharm in apt cache; creating an empty charm instead.
Symlink all hooks to one python source file? [yN] y
INFO:root:Loading charm helper config from charm-helpers.yaml.
INFO:root:Checking out lp:charm-helpers to /tmp/tmpPAqUyN/charm-helpers.
Branched 160 revisions.
INFO:root:Syncing directory: /tmp/tmpPAqUyN/charm-helpers/charmhelpers/core -> lib/
↪charmhelpers/core.
INFO:root:Adding missing __init__.py: lib/charmhelpers/__init__.py
```

Let's see what our new charm looks like:

```
$ tree mycharm/
mycharm/
├── charm-helpers.yaml
└── config.yaml
```

(continues on next page)

(continued from previous page)



The charmhelpers code is bundled in our charm in the `lib/` directory. All of our python code will go in `hooks/hook.py`. A look at that file reveals that charmhelpers has been added to the python path and imported for us:

```

$ head mycharm/hooks/hooks.py -n11
#!/usr/bin/python

import os
import sys

sys.path.insert(0, os.path.join(os.environ['CHARM_DIR'], 'lib'))

from charmhelpers.core import (
    hookenv,
    host,
)

```

### 1.3 Updating Charmhelpers Packages

By default, a new charm installs only the `charmhelpers.core` package, but other packages are available (for a complete list, see the [API Documentation](#)). The installed packages are controlled by the `charm-helpers.yaml` file in our charm:

```

$ cd mycharm
$ cat charm-helpers.yaml
destination: lib/charmhelpers
branch: lp:charm-helpers

```

(continues on next page)



(continued from previous page)

```
include:
  - core
```

Let's update this file to include some more packages:

```
$ vim charm-helpers.yaml
$ cat charm-helpers.yaml
destination: lib/charmhelpers
branch: lp:charm-helpers
include:
  - core
  - contrib.storage
  - fetch
```

Now we need to download the new packages into our charm:

```
$ ./scripts/charm_helpers_sync.py -c charm-helpers.yaml
INFO:root:Loading charm helper config from charm-helpers.yaml.
INFO:root:Checking out lp:charm-helpers to /tmp/tmpT38Y87/charm-helpers.
Branched 160 revisions.
INFO:root:Syncing directory: /tmp/tmpT38Y87/charm-helpers/charmhelpers/core -> lib/
↳charmhelpers/core.
INFO:root:Syncing directory: /tmp/tmpT38Y87/charm-helpers/charmhelpers/contrib/
↳storage -> lib/charmhelpers/contrib/storage.
INFO:root:Adding missing __init__.py: lib/charmhelpers/contrib/__init__.py
INFO:root:Syncing directory: /tmp/tmpT38Y87/charm-helpers/charmhelpers/fetch -> lib/
↳charmhelpers/fetch.
```

A look at our charmhelpers directory reveals that the new packages have indeed been added. We are now free to import and use them in our charm:

```
$ tree lib/charmhelpers/
lib/charmhelpers/
├── contrib
│   ├── __init__.py
│   └── storage
│       ├── __init__.py
│       └── linux
│           ├── ceph.py
│           ├── __init__.py
│           ├── loopback.py
│           ├── lvm.py
│           └── utils.py
├── core
│   ├── fstab.py
│   ├── hookenv.py
│   ├── host.py
│   └── __init__.py
├── fetch
│   ├── archiveurl.py
│   ├── bzrurl.py
│   └── __init__.py
└── __init__.py

5 directories, 15 files
```

## 1.4 Next Steps

Now that you have access to `charmhelpers` in your charm, check out the [Examples](#) or [API Documentation](#) to learn about all the great functionality that `charmhelpers` provides.

If you'd like to contribute an example (please do!), please refer to the *Contributing* page for instructions on how to do so.

## 2.1 Interacting with Charm Configuration

The `charmhelpers.core.hookenv.config()`, when called with no arguments, returns a `charmhelpers.core.hookenv.Config` instance - a dictionary representation of a charm's `config.yaml` file. This object can be used to:

- get a charm's current config values
- check if a config value has changed since the last hook invocation
- view the previous value of a changed config item
- save arbitrary key/value data for use in a later hook

For the following examples we'll assume our charm has a `config.yaml` file that looks like this:

```
options:
  app-name:
    type: string
    default: "My App"
    description: "Name of your app."
```

### 2.1.1 Getting charm config values

```
# hooks/hooks.py

from charmhelpers.core import hookenv

hooks = hookenv.Hooks()
```

(continues on next page)

(continued from previous page)

```
@hooks.hook('install')
def install():
    config = hookenv.config()

    assert config['app-name'] == 'My App'
```

## 2.1.2 Checking if a config value has changed

Let's say the user changes the `app-name` config value at runtime by executing the following juju command:

```
juju set mycharm app-name="My New App"
```

which triggers a `config-changed` hook:

```
# hooks/hooks.py

from charmhelpers.core import hookenv

hooks = hookenv.Hooks()

@hooks.hook('config-changed')
def config_changed():
    config = hookenv.config()

    assert config.changed('app-name')
    assert config['app-name'] == 'My New App'
    assert config.previous('app-name') == 'My App'
```

## 2.1.3 Saving arbitrary key/value data

The `Config` object may be used to store arbitrary data that you want to persist across hook invocations:

```
# hooks/hooks.py

from charmhelpers.core import hookenv

hooks = hookenv.Hooks()

@hooks.hook('install')
def install():
    config = hookenv.config()

    config['mykey'] = 'myval'

@hooks.hook('config-changed')
def config_changed():
    config = hookenv.config()

    assert config['mykey'] == 'myval'
```

## 2.2 Managing Charms with the Services Framework

Traditional charm authoring is focused on implementing hooks. That is, the charm author is thinking in terms of “What hook am I handling; what does this hook need to do?” However, in most cases, the real question should be “Do I have the information I need to configure and start this piece of software and, if so, what are the steps for doing so?” The services framework tries to bring the focus to the data and the setup tasks, in the most declarative way possible.

### 2.2.1 Hooks as Data Sources for Service Definitions

While the `install`, `start`, and `stop` hooks clearly represent state transitions, all of the other hooks are really notifications of changes in data from external sources, such as config data values in the case of `config-changed` or relation data for any of the `*-relation-*` hooks. Moreover, many charms that rely on external data from config options or relations find themselves needing some piece of external data before they can even configure and start anything, and so the `start` hook loses its semantic usefulness.

If data is required from multiple sources, it even becomes impossible to know which hook will be executing when all required data is available. (E.g., which relation will be the last to execute; will the required config option be set before or after all of the relations are available?) One common solution to this problem is to create “flag files” to track whether a given bit of data has been observed, but this can get cluttered quickly and is difficult to understand what conditions lead to which actions.

When using the services framework, all hooks other than `install` are handled by a single call to `manager.manage()`. This can be done with symlinks, or by having a `definitions.py` file containing the service definitions, and every hook can be reduced to:

```
#!/bin/env python
from charmhelpers.core.services import ServiceManager
from definitions import service_definitions
ServiceManager(service_definitions).manage()
```

So, what magic goes into `definitions.py`?

### 2.2.2 Service Definitions Overview

The format of service definitions are fully documented in `ServiceManager`, but most commonly will consist of one or more dictionaries containing four items: the name of a service being managed, the list of data contexts required before the service can be configured and started, the list of actions to take when the data requirements are satisfied, and list of ports to open. The service name generally maps to an Upstart job, the required data contexts are `dict` or `dict`-like structures that contain the data once available (usually subclasses of `RelationContext` or wrappers around `charmhelpers.core.hookenv.config()`), and the actions are just callbacks that are passed the service name for which they are executing (or a subclass of `ManagerCallback` for more complex cases).

An example service definition might be:

```
service_definitions = [
    {
        'service': 'wordpress',
        'ports': [80],
        'required_data': [config(), MySQLRelation()],
        'data_ready': [
            actions.install_frontend,
            services.render_template(source='wp-config.php.j2',
                                   target=os.path.join(WP_INSTALL_DIR, 'wp-config.
↳php'))
```

(continues on next page)

(continued from previous page)

```

        services.render_template(source='wordpress.upstart.j2',
                                target='/etc/init/wordpress'),
    ],
},
]

```

Each time a hook is fired, the conditions will be checked (in this case, just that MySQL is available) and, if met, the appropriate actions taken (correct front-end installed, config files written / updated, and the Upstart job (re)started, implicitly).

## 2.2.3 Required Data Contexts

Required data contexts are, at the most basic level, are just dictionaries, and if they evaluate as True (e.g., if the contain data), their condition is considered to be met. A simple sentinel could just be a function that returns data if available or an empty dict otherwise.

For the common case of gathering data from relations, the *RelationContext* base class gathers data from a named relation and checks for a set of required keys to be present and set on the relation before considering that relation complete. For example, a basic MySQL context might be:

```

class MySQLRelation(RelationContext):
    name = 'db'
    interface = 'mysql'
    required_keys = ['host', 'user', 'password', 'database']

```

Because there could potentially be multiple units on a given relation, and to prevent conflicts when the data contexts are merged to be sent to templates (see below), the data for a *RelationContext* is nested in the following way:

```

relation[relation.name][unit_number][relation_key]

```

For example, to get the host of the first MySQL unit (mysql/0):

```

mysql = MySQLRelation()
unit_0_host = mysql[mysql.name][0]['host']

```

Note that only units that have set values for all of the required keys are included in the list, and if no units have set all of the required keys, instantiating the *RelationContext* will result in an empty list.

## 2.2.4 Data-Ready Actions

When a hook is triggered and all of the *required\_data* contexts are complete, the list of “data ready” actions are executed. These callbacks are passed the service name from the *service* key of the service definition for which they are running, and are responsible for (re)configuring the service according to the required data.

The most common action should be to render a config file from a template. The *render\_template* helper will merge all of the *required\_data* contexts and render a Jinja2 template with the combined data. For example, to render a list of DSNs for units on the db relation, the template should include:

```

databases: [
  {% for unit in db %}
    "mysql://{{unit['user']}}:{{unit['password']}}@{{unit['host']}}/{{unit['database']}}",
  {% endfor %}
]

```

Note that the actions need to be idempotent, since they will all be re-run if something about the charm changes (that is, if a hook is triggered). That is why rendering a template is preferred to editing a file via regular expression substitutions.

Also note that the actions are not responsible for starting the service; there are separate `start` and `stop` options that default to starting and stopping an Upstart service with the name given by the `service` value.

## 2.2.5 Conclusion

By using this framework, it is easy to see what the preconditions for the charm are, and there is never a concern about things being in a partially configured state. As a charm author, you can focus on what is important to you: what data is mandatory, what is optional, and what actions should be taken once the requirements are met.





## 3.1 charmhelpers.core package

### 3.1.1 charmhelpers.core.decorators

`charmhelpers.core.decorators.retry_on_exception` (*num\_retries*, *base\_delay=0*,  
*exc\_type=<class 'Exception'>*)

If the decorated function raises exception *exc\_type*, allow *num\_retries* retry attempts before raise the exception.

### 3.1.2 charmhelpers.core.fstab

**class** `charmhelpers.core.fstab.Fstab` (*path=None*)

Bases: `_io.FileIO`

This class extends file in order to implement a file reader/writer for file `/etc/fstab`

**DEFAULT\_PATH** = `'/etc/fstab'`

**class** `Entry` (*device, mountpoint, filesystem, options, d=0, p=0*)

Bases: `object`

Entry class represents a non-comment line on the `/etc/fstab` file

**classmethod** `add` (*device, mountpoint, filesystem, options=None, path=None*)

**add\_entry** (*entry*)

**entries**

**get\_entry\_by\_attr** (*attr, value*)

**classmethod** `remove_by_mountpoint` (*mountpoint, path=None*)

**remove\_entry** (*entry*)

### 3.1.3 charmhelpers.core.hookenv

<i>Config</i>	A dictionary representation of the charm's config.yaml, with some extra features:
<i>Hooks</i>	A convenient handler for hook functions.
<i>NoNetworkBinding</i>	
<i>Serializable</i>	Wrapper, an object that can be serialized to yaml or json
<i>UnregisteredHookError</i>	Raised when an undefined hook is called
<i>action_fail</i>	Sets the action status to failed and sets the error message.
<i>action_get</i>	Gets the value of an action parameter, or all key/value param pairs
<i>action_name</i>	Get the name of the currently executing action.
<i>action_set</i>	Sets the values to be returned after the action finishes
<i>action_tag</i>	Get the tag for the currently executing action.
<i>action_uuid</i>	Get the UUID of the currently executing action.
<i>add_metric</i>	Add metric values.
<i>application_name</i>	The name of the deployed application this unit belongs to.
<i>application_version_set</i>	Charm authors may trigger this command from any hook to output what version of the application is running.
<i>atexit</i>	Schedule a callback to run on successful hook completion.
<i>atstart</i>	Schedule a callback to run before the main hook.
<i>cached</i>	Cache return values for multiple executions of func + args
<i>charm_dir</i>	Return the root directory of the current charm
<i>charm_name</i>	Get the name of the current charm as is specified on metadata.yaml
<i>close_port</i>	Close a service network port
<i>close_ports</i>	Close a range of service network ports
<i>config</i>	Get the juju charm configuration (scope==None) or individual key, (scope=str).
<i>egress_subnets</i>	Retrieve the egress-subnets from a relation.
<i>env_proxy_settings</i>	Get proxy settings from process environment variables.
<i>execution_environment</i>	A convenient bundling of the current execution context
<i>expected_peer_units</i>	Get a generator for units we expect to join peer relation based on goal-state.
<i>expected_related_units</i>	Get a generator for units we expect to join relation based on goal-state.
<i>flush</i>	Flushes any entries from function cache where the key is found in the function+args
<i>goal_state</i>	Juju goal state values
<i>has_juju_version</i>	Return True if the Juju version is at least the provided version
<i>hook_name</i>	The name of the currently executing hook
<i>in_relation_hook</i>	Determine whether we're running in a relation hook
<i>ingress_address</i>	Retrieve the ingress-address from a relation when available.

Continued on next page

Table 1 – continued from previous page

<i>interface_to_relations</i>	Given an interface, return a list of relation names for the current charm that use that interface.
<i>is_leader</i>	Does the current unit hold the juju leadership
<i>is_relation_made</i>	Determine whether a relation is established by checking for presence of key(s).
<i>iter_units_for_relation_name</i>	Iterate through all units in a relation
<i>juju_version</i>	Full version string (eg.
<i>leader_get</i>	Juju leader get value(s)
<i>leader_set</i>	Juju leader set value(s)
<i>local_unit</i>	Local unit ID
<i>log</i>	Write a message to the juju log
<i>metadata</i>	Get the current charm metadata.yaml contents as a python object
<i>meter_info</i>	Get the meter status information, if running in the meter-status-changed hook.
<i>meter_status</i>	Get the meter status, if running in the meter-status-changed hook.
<i>model_name</i>	Name of the model that this unit is deployed in.
<i>model_uuid</i>	UUID of the model that this unit is deployed in.
<i>network_get</i>	Retrieve the network details for a relation endpoint
<i>network_get_primary_address</i>	Deprecated since Juju 2.3; use <code>network_get()</code>
<i>open_port</i>	Open a service network port
<i>open_ports</i>	Opens a range of service network ports
<i>opened_ports</i>	Get the opened ports
<i>payload_register</i>	is used while a hook is running to let Juju know that a payload has been started.
<i>payload_status_set</i>	is used to update the current status of a registered payload.
<i>payload_unregister</i>	is used while a hook is running to let Juju know that a payload has been manually stopped.
<i>peer_relation_id</i>	Get the peers relation id if a peers relation has been joined, else None.
<i>principal_unit</i>	Returns the principal unit of this unit, otherwise None
<i>related_units</i>	A list of related units
<i>relation_clear</i>	Clears any relation data already set on relation <code>r_id</code>
<i>relation_for_unit</i>	Get the json representation of a unit's relation
<i>relation_get</i>	Get relation information
<i>relation_id</i>	The relation ID for the current or a specified relation
<i>relation_ids</i>	A list of relation_ids
<i>relation_set</i>	Set relation information for the current unit
<i>relation_to_interface</i>	Given the name of a relation, return the interface that relation uses.
<i>relation_to_role_and_interface</i>	Given the name of a relation, return the role and the name of the interface that relation uses (where role is one of <code>provides</code> , <code>requires</code> , or <code>peers</code> ).
<i>relation_type</i>	The scope for the current relation hook
<i>relation_types</i>	Get a list of relation types supported by this charm
<i>relations</i>	Get a nested dictionary of relation data for all related units
<i>relations_for_id</i>	Get relations of a specific relation ID
<i>relations_of_type</i>	Get relations of a specific type

Continued on next page

Table 1 – continued from previous page

<code>remote_service_name</code>	The remote service name for a given relation-id (or the current relation)
<code>remote_unit</code>	The remote unit for the current relation hook
<code>resource_get</code>	used to fetch the resource path of the given name.
<code>role_and_interface_to_relations</code>	Given a role and interface name, return a list of relation names for the current charm that use that interface under that role (where role is one of <code>provides</code> , <code>requires</code> , or <code>peers</code> ).
<code>service_name</code>	Deprecated since version 0.19.1.
<code>status_get</code>	Retrieve the previously set juju workload state and message
<code>status_set</code>	Set the workload state with a message
<code>storage_get</code>	Get storage attributes
<code>storage_list</code>	List the storage IDs for the unit
<code>translate_exc</code>	
<code>unit_doomed</code>	Determines if the unit is being removed from the model
<code>unit_get</code>	Get the unit ID for the remote unit
<code>unit_private_ip</code>	Get this unit's private IP address
<code>unit_public_ip</code>	Get this unit's public IP address

### Interactions with the Juju environment

**class** `charmhelpers.core.hookenv.Config` (\*args, \*\*kw)

Bases: dict

A dictionary representation of the charm's config.yaml, with some extra features:

- See which values in the dictionary have changed since the previous hook.
- For values that have changed, see what the previous value was.
- Store arbitrary data for use in a later hook.

NOTE: Do not instantiate this object directly - instead call `hookenv.config()`, which will return an instance of `Config`.

Example usage:

```
>>> # inside a hook
>>> from charmhelpers.core import hookenv
>>> config = hookenv.config()
>>> config['foo']
'bar'
>>> # store a new key/value for later use
>>> config['mykey'] = 'myval'

>>> # user runs `juju set mycharm foo=baz`
>>> # now we're inside subsequent config-changed hook
>>> config = hookenv.config()
>>> config['foo']
'baz'
>>> # test to see if this val has changed since last hook
>>> config.changed('foo')
True
>>> # what was the previous value?
```

(continues on next page)

(continued from previous page)

```
>>> config.previous('foo')
'bar'
>>> # keys/values that we add are preserved across hooks
>>> config['mykey']
'myval'
```

**CONFIG\_FILE\_NAME** = '.juju-persistent-config'

**changed** (*key*)

Return True if the current value for this key is different from the previous value.

**load\_previous** (*path=None*)

Load previous copy of config from disk.

In normal usage you don't need to call this method directly - it is called automatically at object initialization.

**Parameters path** – File path from which to load the previous config. If *None*, config is loaded from the default location. If *path* is specified, subsequent *save()* calls will write to the same path.

**previous** (*key*)

Return previous value for this key, or *None* if there is no previous value.

**save** ()

Save this config to disk.

If the charm is using the `Services Framework` or `:meth:'@hook <Hooks.hook>'` decorator, this is called automatically at the end of successful hook execution. Otherwise, it should be called directly by user code.

To disable automatic saves, set `implicit_save=False` on this instance.

**class** `charmhelpers.core.hookenv.Hooks` (*config\_save=None*)

Bases: object

A convenient handler for hook functions.

Example:

```
hooks = Hooks()

# register a hook, taking its name from the function name
@hooks.hook()
def install():
    pass # your code here

# register a hook, providing a custom hook name
@hooks.hook("config-changed")
def config_changed():
    pass # your code here

if __name__ == "__main__":
    # execute a hook based on the name the program is called by
    hooks.execute(sys.argv)
```

**execute** (*args*)

Execute a registered hook based on `args[0]`

**hook** (*\*hook\_names*)

Decorator, registering them as hooks

**register** (*name, function*)

Register a hook

**exception** `charmhelpers.core.hookenv.NoNetworkBinding`

Bases: `Exception`

**class** `charmhelpers.core.hookenv.Serializable` (*obj*)

Bases: `collections.UserDict`

Wrapper, an object that can be serialized to yaml or json

**json** ()

Serialize the object to json

**yaml** ()

Serialize the object to yaml

**exception** `charmhelpers.core.hookenv.UnregisteredHookError`

Bases: `Exception`

Raised when an undefined hook is called

`charmhelpers.core.hookenv.action_fail` (*message*)

Sets the action status to failed and sets the error message.

The results set by `action_set` are preserved.

`charmhelpers.core.hookenv.action_get` (*key=None*)

Gets the value of an action parameter, or all key/value param pairs

`charmhelpers.core.hookenv.action_name` ()

Get the name of the currently executing action.

`charmhelpers.core.hookenv.action_set` (*values*)

Sets the values to be returned after the action finishes

`charmhelpers.core.hookenv.action_tag` ()

Get the tag for the currently executing action.

`charmhelpers.core.hookenv.action_uuid` ()

Get the UUID of the currently executing action.

`charmhelpers.core.hookenv.add_metric` (*\*args, \*\*kwargs*)

Add metric values. Values may be expressed with keyword arguments. For metric names containing dashes, these may be expressed as one or more 'key=value' positional arguments. May only be called from the collect-metrics hook.

`charmhelpers.core.hookenv.application_name` ()

The name of the deployed application this unit belongs to.

`charmhelpers.core.hookenv.application_version_set` (*version*)

Charm authors may trigger this command from any hook to output what version of the application is running. This could be a package version, for instance postgres version 9.5. It could also be a build number or version control revision identifier, for instance git sha 6fb7ba68.

`charmhelpers.core.hookenv.atexit` (*callback, \*args, \*\*kwargs*)

Schedule a callback to run on successful hook completion.

Callbacks are run in the reverse order that they were added.

`charmhelpers.core.hookenv.atstart` (*callback, \*args, \*\*kwargs*)

Schedule a callback to run before the main hook.

Callbacks are run in the order they were added.

This is useful for modules and classes to perform initialization and inject behavior. In particular:

- Run common code before all of your hooks, such as logging the hook name or interesting relation data.
- Defer object or module initialization that requires a hook context until we know there actually is a hook context, making testing easier.
- Rather than requiring charm authors to include boilerplate to invoke your helper's behavior, have it run automatically if your object is instantiated or module imported.

This is not at all useful after your hook framework as been launched.

`charmhelpers.core.hookenv.cached` (*func*)

Cache return values for multiple executions of `func + args`

For example:

```
@cached
def unit_get(attribute):
    pass

unit_get('test')
```

will cache the result of `unit_get + 'test'` for future calls.

`charmhelpers.core.hookenv.charm_dir` ()

Return the root directory of the current charm

`charmhelpers.core.hookenv.charm_name` ()

Get the name of the current charm as is specified on `metadata.yaml`

`charmhelpers.core.hookenv.close_port` (*port, protocol='TCP'*)

Close a service network port

`charmhelpers.core.hookenv.close_ports` (*start, end, protocol='TCP'*)

Close a range of service network ports

`charmhelpers.core.hookenv.config` (*scope=None*)

Get the juju charm configuration (`scope==None`) or individual key, (`scope=str`). The returned value is a Python data structure loaded as JSON from the Juju config command.

**Parameters** `scope` (*Optional[str]*) – If set, return the value for the specified key.

**Returns** Either the whole config as a `Config`, or a key from it.

**Return type** Any

`charmhelpers.core.hookenv.egress_subnets` (*rid=None, unit=None*)

Retrieve the egress-subnets from a relation.

This function is to be used on the providing side of the relation, and provides the ranges of addresses that client connections may come from. The result is uninteresting on the consuming side of a relation (`unit == local_unit()`).

Returns a stable list of subnets in CIDR format. eg. [`'192.168.1.0/24'`, `'2001::F00F/128'`]

If egress-subnets is not available, falls back to using the published ingress-address, or finally private-address.

**Parameters**

- `rid` – string relation id
- `unit` – string unit name

**Side effect** calls `relation_get`

**Returns** list of subnets in CIDR format. eg. ['192.168.1.0/24', '2001::F00F/128']

`charmhelpers.core.hookenv.env_proxy_settings` (*selected\_settings=None*)

Get proxy settings from process environment variables.

Get charm proxy settings from environment variables that correspond to juju-http-proxy, juju-https-proxy and juju-no-proxy (available as of 2.4.2, see lp:1782236) in a format suitable for passing to an application that reacts to proxy settings passed as environment variables. Some applications support lowercase or uppercase notation (e.g. curl), some support only lowercase (e.g. wget), there are also subjectively rare cases of only uppercase notation support. no\_proxy CIDR and wildcard support also varies between runtimes and applications as there is no enforced standard.

Some applications may connect to multiple destinations and expose config options that would affect only proxy settings for a specific destination these should be handled in charms in an application-specific manner.

**Parameters** `selected_settings` (*list*) – format only a subset of possible settings

**Return type** Option(None, dict[str, str])

`charmhelpers.core.hookenv.execution_environment` ()

A convenient bundling of the current execution context

`charmhelpers.core.hookenv.expected_peer_units` ()

Get a generator for units we expect to join peer relation based on goal-state.

The local unit is excluded from the result to make it easy to gauge completion of all peers joining the relation with existing hook tools.

Example usage: `log('peer {} of {} joined peer relation')`

`.format(len(related_units()), len(list(expected_peer_units())))`

This function will raise `NotImplementedError` if used with juju versions without goal-state support.

**Returns** iterator

**Return type** types.GeneratorType

**Raises** `NotImplementedError`

`charmhelpers.core.hookenv.expected_related_units` (*reltype=None*)

Get a generator for units we expect to join relation based on goal-state.

Note that you can not use this function for the peer relation, take a look at `expected_peer_units()` for that.

This function will raise `KeyError` if you request information for a relation type for which juju goal-state does not have information. It will raise `NotImplementedError` if used with juju versions without goal-state support.

Example usage: `log('participant {} of {} joined relation {}')`

`.format(len(related_units()), len(list(expected_related_units())), relation_type())`

**Parameters** `reltype` (*str*) – Relation type to list data for, default is to list data for the relation type we are currently executing a hook for.

**Returns** iterator

**Return type** types.GeneratorType

**Raises** `KeyError`, `NotImplementedError`

`charmhelpers.core.hookenv.flush` (*key*)

Flushes any entries from function cache where the key is found in the function+args



`charmhelpers.core.hookenv.goal_state()`

Juju goal state values

`charmhelpers.core.hookenv.has_juju_version(minimum_version)`

Return True if the Juju version is at least the provided version

`charmhelpers.core.hookenv.hook_name()`

The name of the currently executing hook

`charmhelpers.core.hookenv.in_relation_hook()`

Determine whether we're running in a relation hook

`charmhelpers.core.hookenv.ingress_address(rid=None, unit=None)`

Retrieve the ingress-address from a relation when available. Otherwise, return the private-address.

When used on the consuming side of the relation (`unit` is a remote unit), the ingress-address is the IP address that this unit needs to use to reach the provided service on the remote unit.

When used on the providing side of the relation (`unit == local_unit()`), the ingress-address is the IP address that is advertised to remote units on this relation. Remote units need to use this address to reach the local provided service on this unit.

Note that charms may document some other method to use in preference to the `ingress_address()`, such as an address provided on a different relation attribute or a service discovery mechanism. This allows charms to redirect inbound connections to their peers or different applications such as load balancers.

Usage: `addresses = [ingress_address(rid=u.rid, unit=u.unit)`

`for u in iter_units_for_relation_name(relation_name)]`

#### Parameters

- **rid** – string relation id
- **unit** – string unit name

**Side effect** calls `relation_get`

**Returns** string IP address

`charmhelpers.core.hookenv.interface_to_relations(interface_name)`

Given an interface, return a list of relation names for the current charm that use that interface.

**Returns** A list of relation names.

`charmhelpers.core.hookenv.is_leader()`

Does the current unit hold the juju leadership

Uses juju to determine whether the current unit is the leader of its peers

`charmhelpers.core.hookenv.is_relation_made(relation, keys='private-address')`

Determine whether a relation is established by checking for presence of key(s). If a list of keys is provided, they must all be present for the relation to be identified as made

`charmhelpers.core.hookenv.iter_units_for_relation_name(relation_name)`

Iterate through all units in a relation

Generator that iterates through all the units in a relation and yields a named tuple with `rid` and `unit` field names.

Usage: `data = [(u.rid, u.unit)`

`for u in iter_units_for_relation_name(relation_name)]`

**Parameters** `relation_name` – string relation name

**Yield** Named Tuple with rid and unit field names

`charmhelpers.core.hookenv.juju_version()`  
Full version string (eg. '1.23.3.1-trusty-amd64')

`charmhelpers.core.hookenv.leader_get(attribute=None)`  
Juju leader get value(s)

`charmhelpers.core.hookenv.leader_set(settings=None, **kwargs)`  
Juju leader set value(s)

`charmhelpers.core.hookenv.local_unit()`  
Local unit ID

`charmhelpers.core.hookenv.log(message, level=None)`  
Write a message to the juju log

`charmhelpers.core.hookenv.metadata()`  
Get the current charm metadata.yaml contents as a python object

`charmhelpers.core.hookenv.meter_info()`  
Get the meter status information, if running in the meter-status-changed hook.

`charmhelpers.core.hookenv.meter_status()`  
Get the meter status, if running in the meter-status-changed hook.

`charmhelpers.core.hookenv.model_name()`  
Name of the model that this unit is deployed in.

`charmhelpers.core.hookenv.model_uuid()`  
UUID of the model that this unit is deployed in.

`charmhelpers.core.hookenv.network_get(endpoint, relation_id=None)`  
Retrieve the network details for a relation endpoint

**Parameters**

- **endpoint** – string. The name of a relation endpoint
- **relation\_id** – int. The ID of the relation for the current context.

**Returns** dict. The loaded YAML output of the network-get query.

**Raise** `NotImplementedError` if request not supported by the Juju version.

`charmhelpers.core.hookenv.network_get_primary_address(binding)`  
Deprecated since Juju 2.3; use `network_get()`

Retrieve the primary network address for a named binding

**Parameters** **binding** – string. The name of a relation of extra-binding

**Returns** string. The primary IP address for the named binding

**Raise** `NotImplementedError` if run on Juju < 2.0

`charmhelpers.core.hookenv.open_port(port, protocol='TCP')`  
Open a service network port

`charmhelpers.core.hookenv.open_ports(start, end, protocol='TCP')`  
Opens a range of service network ports

`charmhelpers.core.hookenv.opened_ports()`  
Get the opened ports

*Note that this will only show ports opened in a previous hook*

**Returns** Opened ports as a list of strings: ['8080/tcp', '8081-8083/tcp']

`charmhelpers.core.hookenv.payload_register` (*ptype, klass, pid*)

is used while a hook is running to let Juju know that a payload has been started.

`charmhelpers.core.hookenv.payload_status_set` (*klass, pid, status*)

is used to update the current status of a registered payload. The <class> and <id> provided must match a payload that has been previously registered with juju using payload-register. The <status> must be one of the follow: starting, started, stopping, stopped

`charmhelpers.core.hookenv.payload_unregister` (*klass, pid*)

is used while a hook is running to let Juju know that a payload has been manually stopped. The <class> and <id> provided must match a payload that has been previously registered with juju using payload-register.

`charmhelpers.core.hookenv.peer_relation_id` ()

Get the peers relation id if a peers relation has been joined, else None.

`charmhelpers.core.hookenv.principal_unit` ()

Returns the principal unit of this unit, otherwise None

`charmhelpers.core.hookenv.related_units` (*relid=None*)

A list of related units

`charmhelpers.core.hookenv.relation_clear` (*r\_id=None*)

Clears any relation data already set on relation r\_id

`charmhelpers.core.hookenv.relation_for_unit` (*unit=None, rid=None*)

Get the json representation of a unit's relation

`charmhelpers.core.hookenv.relation_get` (*attribute=None, unit=None, rid=None*)

Get relation information

`charmhelpers.core.hookenv.relation_id` (*relation\_name=None, service\_or\_unit=None*)

The relation ID for the current or a specified relation

`charmhelpers.core.hookenv.relation_ids` (*reltype=None*)

A list of relation\_ids

`charmhelpers.core.hookenv.relation_set` (*relation\_id=None, relation\_settings=None, \*\*kwargs*)

Set relation information for the current unit

`charmhelpers.core.hookenv.relation_to_interface` (*relation\_name*)

Given the name of a relation, return the interface that relation uses.

**Returns** The interface name, or None.

`charmhelpers.core.hookenv.relation_to_role_and_interface` (*relation\_name*)

Given the name of a relation, return the role and the name of the interface that relation uses (where role is one of provides, requires, or peers).

**Returns** A tuple containing (role, interface), or (None, None).

`charmhelpers.core.hookenv.relation_type` ()

The scope for the current relation hook

`charmhelpers.core.hookenv.relation_types` ()

Get a list of relation types supported by this charm

`charmhelpers.core.hookenv.relations` ()

Get a nested dictionary of relation data for all related units

`charmhelpers.core.hookenv.relations_for_id` (*relid=None*)

Get relations of a specific relation ID

`charmhelpers.core.hookenv.relations_of_type` (*reltype=None*)  
Get relations of a specific type

`charmhelpers.core.hookenv.remote_service_name` (*relid=None*)  
The remote service name for a given relation-id (or the current relation)

`charmhelpers.core.hookenv.remote_unit` ()  
The remote unit for the current relation hook

`charmhelpers.core.hookenv.resource_get` (*name*)  
used to fetch the resource path of the given name.  
  
<name> must match a name of defined resource in metadata.yaml  
  
returns either a path or False if resource not available

`charmhelpers.core.hookenv.role_and_interface_to_relations` (*role, interface\_name*)  
Given a role and interface name, return a list of relation names for the current charm that use that interface under that role (where role is one of provides, requires, or peers).  
  
**Returns** A list of relation names.

`charmhelpers.core.hookenv.service_name` ()  
Deprecated since version 0.19.1: Alias for `application_name` ().

`charmhelpers.core.hookenv.status_get` ()  
Retrieve the previously set juju workload state and message  
  
If the status-get command is not found then assume this is juju < 1.23 and return 'unknown', ""

`charmhelpers.core.hookenv.status_set` (*workload\_state, message*)  
Set the workload state with a message  
  
Use status-set to set the workload state with a message which is visible to the user via juju status. If the status-set command is not found then assume this is juju < 1.23 and juju-log the message instead.  
  
workload\_state – valid juju workload state. message – status update message

`charmhelpers.core.hookenv.storage_get` (*attribute=None, storage\_id=None*)  
Get storage attributes

`charmhelpers.core.hookenv.storage_list` (*storage\_name=None*)  
List the storage IDs for the unit

`charmhelpers.core.hookenv.translate_exc` (*from\_exc, to\_exc*)

`charmhelpers.core.hookenv.unit_doomed` (*unit=None*)  
Determines if the unit is being removed from the model  
  
Requires Juju 2.4.1.

**Parameters** `unit` – string unit name, defaults to `local_unit`

**Side effect** calls `goal_state`

**Side effect** calls `local_unit`

**Side effect** calls `has_juju_version`

**Returns** True if the unit is being removed, already gone, or never existed

`charmhelpers.core.hookenv.unit_get` (*attribute*)  
Get the unit ID for the remote unit

`charmhelpers.core.hookenv.unit_private_ip` ()  
Get this unit's private IP address

`charmhelpers.core.hookenv.unit_public_ip()`  
 Get this unit's public IP address

### 3.1.4 charmhelpers.core.host

<code>ChecksumError</code>	A class derived from Value error to indicate the checksum failed.
<code>CompareHostReleases</code>	Provide comparisons of Ubuntu releases.
<code>add_group</code>	Add a group to the system
<code>add_new_group</code>	
<code>add_to_updatedb_prunepath</code>	Adds the specified path to the mlocate's updatedb.conf PRUNEPATH list.
<code>add_user_to_group</code>	Add a user to a group
<code>adduser</code>	Add a user to the system.
<code>arch</code>	Return the package architecture as a string.
<code>chage</code>	Change user password expiry information
<code>chdir</code>	Change the current working directory to a different directory for a code block and return the previous directory after the block exits.
<code>check_hash</code>	Validate a file using a cryptographic checksum.
<code>chownr</code>	Recursively change user and group ownership of files and directories in given path.
<code>cmp_pkgrevno</code>	Compare supplied revno with the revno of the installed package.
<code>file_hash</code>	Generate a hash checksum of the contents of 'path' or None if not found.
<code>fstab_add</code>	Adds the given device entry to the /etc/fstab file
<code>fstab_mount</code>	Mount filesystem using fstab
<code>fstab_remove</code>	Remove the given mountpoint entry from /etc/fstab
<code>get_bond_master</code>	Returns bond master if interface is bond slave otherwise None.
<code>get_distrib_codename</code>	Return the codename of the distribution :returns: The codename :rtype: str
<code>get_nic_hwaddr</code>	Return the Media Access Control (MAC) for a network interface.
<code>get_nic_mtu</code>	Return the Maximum Transmission Unit (MTU) for a network interface.
<code>get_total_ram</code>	The total amount of system RAM in bytes.
<code>gid_exists</code>	Check if a gid exists
<code>group_exists</code>	Check if a group exists
<code>init_is_systemd</code>	Return True if the host system uses systemd, False otherwise.
<code>install_ca_cert</code>	Install the given cert as a trusted CA.
<code>is_container</code>	Determine whether unit is running in a container
<code>is_phy_iface</code>	Returns True if interface is not virtual, otherwise False.
<code>lchownr</code>	Recursively change user and group ownership of files and directories in a given path, not following symbolic links.
<code>list_nics</code>	Return a list of nics of given type(s)
<code>lsb_release</code>	Return /etc/lsb-release in a dict

Continued on next page

Table 2 – continued from previous page

<code>mkdir</code>	Create a directory
<code>modulo_distribution</code>	Modulo distribution
<code>mount</code>	Mount a filesystem at a particular mountpoint
<code>mounts</code>	Get a list of all mounted volumes as <code>[[mountpoint,device],[...]]</code>
<code>owner</code>	Returns a tuple containing the username & groupname owning the path.
<code>path_hash</code>	Generate a hash checksum of all files matching 'path'.
<code>pwgen</code>	Generate a random password.
<code>restart_on_change</code>	Restart services based on configuration files changing
<code>restart_on_change_helper</code>	Helper function to perform the restart_on_change function.
<code>rsync</code>	Replicate the contents of a path
<code>service</code>	Control a system service.
<code>service_available</code>	Determine whether a system service is available
<code>service_pause</code>	Pause a system service.
<code>service_reload</code>	Reload a system service, optionally falling back to restart if reload fails.
<code>service_restart</code>	Restart a system service.
<code>service_resume</code>	Resume a system service.
<code>service_running</code>	Determine whether a system service is running.
<code>service_start</code>	Start a system service.
<code>service_stop</code>	Stop a system service.
<code>set_nic_mtu</code>	Set the Maximum Transmission Unit (MTU) on a network interface.
<code>symlink</code>	Create a symbolic link
<code>uid_exists</code>	Check if a uid exists
<code>umount</code>	Unmount a filesystem
<code>updatedb</code>	
<code>user_exists</code>	Check if a user exists
<code>write_file</code>	Create or overwrite a file with the contents of a byte string.

### Tools for working with the host system

**exception** `charmhelpers.core.host.ChecksumError`

Bases: `ValueError`

A class derived from Value error to indicate the checksum failed.

`charmhelpers.core.host.add_group` (*group\_name*, *system\_group=False*, *gid=None*)

Add a group to the system

Will log but otherwise succeed if the group already exists.

#### Parameters

- **group\_name** (*str*) – group to create
- **system\_group** (*bool*) – Create system group
- **gid** (*int*) – GID for user being created

**Returns** The password database entry struct, as returned by `grp.getgrnam`

`charmhelpers.core.host.add_to_updatedb_prunepath` (*path*, *updatedb\_path*='/etc/updatedb.conf')

Adds the specified path to the mlocate's updatedb.conf PRUNEPATH list.

This method has no effect if the path specified by `updatedb_path` does not exist or is not a file.

@param `path`: string the path to add to the updatedb.conf PRUNEPATHS value @param `updatedb_path`: the path the updatedb.conf file

`charmhelpers.core.host.add_user_to_group` (*username*, *group*)

Add a user to a group

`charmhelpers.core.host.adduser` (*username*, *password*=None, *shell*='/bin/bash', *system\_user*=False, *primary\_group*=None, *secondary\_groups*=None, *uid*=None, *home\_dir*=None)

Add a user to the system.

Will log but otherwise succeed if the user already exists.

#### Parameters

- **username** (*str*) – Username to create
- **password** (*str*) – Password for user; if None, create a system user
- **shell** (*str*) – The default shell for the user
- **system\_user** (*bool*) – Whether to create a login or system user
- **primary\_group** (*str*) – Primary group for user; defaults to username
- **secondary\_groups** (*list*) – Optional list of additional groups
- **uid** (*int*) – UID for user being created
- **home\_dir** (*str*) – Home directory for user

**Returns** The password database entry struct, as returned by `pwd.getpwnam`

`charmhelpers.core.host.chage` (*username*, *lastday*=None, *expiredate*=None, *inactive*=None, *mindays*=None, *maxdays*=None, *root*=None, *warndays*=None)

Change user password expiry information

#### Parameters

- **username** (*str*) – User to update
- **lastday** (*str*) – Set when password was changed in YYYY-MM-DD format
- **expiredate** (*str*) – Set when user's account will no longer be accessible in YYYY-MM-DD format. -1 will remove an account expiration date.
- **inactive** (*str*) – Set the number of days of inactivity after a password has expired before the account is locked. -1 will remove an account's inactivity.
- **mindays** (*str*) – Set the minimum number of days between password changes to MIN\_DAYS. 0 indicates the password can be changed anytime.
- **maxdays** (*str*) – Set the maximum number of days during which a password is valid. -1 as MAX\_DAYS will remove checking maxdays
- **root** (*str*) – Apply changes in the CHROOT\_DIR directory
- **warndays** (*str*) – Set the number of days of warning before a password change is required

**Raises** `subprocess.CalledProcessError` – if call to chage fails

`charmhelpers.core.host.chdir` (*directory*)

Change the current working directory to a different directory for a code block and return the previous directory after the block exits. Useful to run commands from a specified directory.

**Parameters** `directory` (*str*) – The directory path to change to for this context.

`charmhelpers.core.host.check_hash` (*path, checksum, hash\_type='md5'*)

Validate a file using a cryptographic checksum.

**Parameters**

- **checksum** (*str*) – Value of the checksum used to validate the file.
- **hash\_type** (*str*) – Hash algorithm used to generate *checksum*. Can be any hash algorithm supported by `hashlib`, such as `md5`, `sha1`, `sha256`, `sha512`, etc.

**Raises** `ChecksumError` – If the file fails the checksum

`charmhelpers.core.host.chownr` (*path, owner, group, follow\_links=True, chowntopdir=False*)

Recursively change user and group ownership of files and directories in given path. Doesn't chown path itself by default, only its children.

**Parameters**

- **path** (*str*) – The string path to start changing ownership.
- **owner** (*str*) – The owner string to use when looking up the uid.
- **group** (*str*) – The group string to use when looking up the gid.
- **follow\_links** (*bool*) – Also follow and chown links if True
- **chowntopdir** (*bool*) – Also chown path itself if True

`charmhelpers.core.host.file_hash` (*path, hash\_type='md5'*)

Generate a hash checksum of the contents of 'path' or None if not found.

**Parameters** `hash_type` (*str*) – Any hash algorithm supported by `hashlib`, such as `md5`, `sha1`, `sha256`, `sha512`, etc.

`charmhelpers.core.host.fstab_add` (*dev, mp, fs, options=None*)

Adds the given device entry to the `/etc/fstab` file

`charmhelpers.core.host.fstab_mount` (*mountpoint*)

Mount filesystem using `fstab`

`charmhelpers.core.host.fstab_remove` (*mp*)

Remove the given mountpoint entry from `/etc/fstab`

`charmhelpers.core.host.get_bond_master` (*interface*)

Returns bond master if interface is bond slave otherwise None.

NOTE: the provided interface is expected to be physical

`charmhelpers.core.host.get_nic_hwaddr` (*nic*)

Return the Media Access Control (MAC) for a network interface.

`charmhelpers.core.host.get_nic_mtu` (*nic*)

Return the Maximum Transmission Unit (MTU) for a network interface.

`charmhelpers.core.host.get_total_ram` ()

The total amount of system RAM in bytes.

This is what is reported by the OS, and may be overcommitted when there are multiple containers hosted on the same machine.



`charmhelpers.core.host.gid_exists (gid)`

Check if a gid exists

`charmhelpers.core.host.group_exists (groupname)`

Check if a group exists

`charmhelpers.core.host.init_is_systemd ()`

Return True if the host system uses systemd, False otherwise.

`charmhelpers.core.host.install_ca_cert (ca_cert, name=None)`

Install the given cert as a trusted CA.

The name is the stem of the filename where the cert is written, and if not provided, it will default to `juju-{charm_name}`.

If the cert is empty or None, or is unchanged, nothing is done.

`charmhelpers.core.host.is_container ()`

Determine whether unit is running in a container

@return: boolean indicating if unit is in a container

`charmhelpers.core.host.is_phy_iface (interface)`

Returns True if interface is not virtual, otherwise False.

`charmhelpers.core.host.lchownr (path, owner, group)`

Recursively change user and group ownership of files and directories in a given path, not following symbolic links. See the documentation for 'os.lchown' for more information.

#### Parameters

- **path** (*str*) – The string path to start changing ownership.
- **owner** (*str*) – The owner string to use when looking up the uid.
- **group** (*str*) – The group string to use when looking up the gid.

`charmhelpers.core.host.list_nics (nic_type=None)`

Return a list of nics of given type(s)

`charmhelpers.core.host.mkdir (path, owner='root', group='root', perms=365, force=False)`

Create a directory

`charmhelpers.core.host.modulo_distribution (modulo=3, wait=30, non_zero_wait=False)`

Modulo distribution

This helper uses the unit number, a modulo value and a constant wait time to produce a calculated wait time distribution. This is useful in large scale deployments to distribute load during an expensive operation such as service restarts.

If you have 1000 nodes that need to restart 100 at a time 1 minute at a time:

```
time.wait(modulo_distribution(modulo=100, wait=60)) restart()
```

If you need restarts to happen serially set modulo to the exact number of nodes and set a high constant wait time:

```
time.wait(modulo_distribution(modulo=10, wait=120)) restart()
```

@param modulo: int The modulo number creates the group distribution @param wait: int The constant time wait value @param non\_zero\_wait: boolean Override unit % modulo == 0,

return modulo \* wait. Used to avoid collisions with leader nodes which are often given priority.

@return: int Calculated time to wait for unit operation

`charmhelpers.core.host.mount` (*device*, *mountpoint*, *options=None*, *persist=False*, *filesystem='ext3'*)  
Mount a filesystem at a particular mountpoint

`charmhelpers.core.host.mounts` ()  
Get a list of all mounted volumes as [[mountpoint,device],[...]]

`charmhelpers.core.host.owner` (*path*)  
Returns a tuple containing the username & groupname owning the path.

**Parameters** *path* (*str*) – the string path to retrieve the ownership

**Return tuple**(*str*, *str*) A (username, groupname) tuple containing the name of the user and group owning the path.

**Raises** **OSError** – if the specified path does not exist

`charmhelpers.core.host.path_hash` (*path*)  
Generate a hash checksum of all files matching 'path'. Standard wildcards like '\*' and '?' are supported, see documentation for the 'glob' module for more information.

**Returns** dict: A { filename: hash } dictionary for all matched files. Empty if none found.

`charmhelpers.core.host.pwgen` (*length=None*)  
Generate a random password.

`charmhelpers.core.host.remove_password_expiry` (*username*, *lastday=None*, *\**,  
*expiredate='-1'*, *inactive='-1'*, *mindays='0'*, *maxdays='-1'*, *root=None*,  
*warndays=None*)

Change user password expiry information

**Parameters**

- **username** (*str*) – User to update
- **lastday** (*str*) – Set when password was changed in YYYY-MM-DD format
- **expiredate** (*str*) – Set when user's account will no longer be accessible in YYYY-MM-DD format. -1 will remove an account expiration date.
- **inactive** (*str*) – Set the number of days of inactivity after a password has expired before the account is locked. -1 will remove an account's inactivity.
- **mindays** (*str*) – Set the minimum number of days between password changes to MIN\_DAYS. 0 indicates the password can be changed anytime.
- **maxdays** (*str*) – Set the maximum number of days during which a password is valid. -1 as MAX\_DAYS will remove checking maxdays
- **root** (*str*) – Apply changes in the CHROOT\_DIR directory
- **warndays** (*str*) – Set the number of days of warning before a password change is required

**Raises** **subprocess.CalledProcessError** – if call to chage fails

`charmhelpers.core.host.restart_on_change` (*restart\_map*, *stopstart=False*,  
*restart\_functions=None*)  
Restart services based on configuration files changing

This function is used a decorator, for example:

```
@restart_on_change({
    '/etc/ceph/ceph.conf': [ 'cinder-api', 'cinder-volume' ]
    '/etc/apache/sites-enabled/*': [ 'apache2' ]
})
def config_changed():
    pass # your code here
```

In this example, the cinder-api and cinder-volume services would be restarted if /etc/ceph/ceph.conf is changed by the ceph\_client\_changed function. The apache2 service would be restarted if any file matching the pattern got changed, created or removed. Standard wildcards are supported, see documentation for the ‘glob’ module for more information.

@param restart\_map: {path\_file\_name: [service\_name, ...] @param stopstart: DEFAULT false; whether to stop, start OR restart @param restart\_functions: nonstandard functions to use to restart services

```
{svc: func, ... }
```

@returns result from decorated function

```
charmhelpers.core.host.restart_on_change_helper(lambda_f, restart_map, stop-
start=False, restart_functions=None)
```

Helper function to perform the restart\_on\_change function.

This is provided for decorators to restart services if files described in the restart\_map have changed after an invocation of lambda\_f().

@param lambda\_f: function to call. @param restart\_map: {file: [service, ...]} @param stopstart: whether to stop, start or restart a service @param restart\_functions: nonstandard functions to use to restart services

```
{svc: func, ... }
```

@returns result of lambda\_f()

```
charmhelpers.core.host.rsync(from_path, to_path, flags='-r', options=None, timeout=None)
```

Replicate the contents of a path

```
charmhelpers.core.host.service(action, service_name, **kwargs)
```

Control a system service.

#### Parameters

- **action** – the action to take on the service
- **service\_name** – the name of the service to perform th action on
- **\*\*kwargs** – additional params to be passed to the service command in the form of key=value.

```
charmhelpers.core.host.service_pause(service_name, init_dir='/etc/init',
initd_dir='/etc/init.d', **kwargs)
```

Pause a system service.

Stop it, and prevent it from starting again at boot.

#### Parameters

- **service\_name** – the name of the service to pause
- **init\_dir** – path to the upstart init directory
- **initd\_dir** – path to the sysv init directory
- **\*\*kwargs** – additional parameters to pass to the init system when managing services. These will be passed as key=value parameters to the init system’s commandline. kwargs are ignored for init systems which do not support key=value arguments via the commandline.

`charmhelpers.core.host.service_reload` (*service\_name*, *restart\_on\_failure=False*, *\*\*kwargs*)

Reload a system service, optionally falling back to restart if reload fails.

The specified service name is managed via the system level init system. Some init systems (e.g. upstart) require that additional arguments be provided in order to directly control service instances whereas other init systems allow for addressing instances of a service directly by name (e.g. systemd).

The kwargs allow for the additional parameters to be passed to underlying init systems for those systems which require/allow for them. For example, the ceph-osd upstart script requires the `id` parameter to be passed along in order to identify which running daemon should be reloaded. The following example restarts the ceph-osd service for instance `id=4`:

```
service_reload('ceph-osd', id=4)
```

### Parameters

- **service\_name** – the name of the service to reload
- **restart\_on\_failure** – boolean indicating whether to fallback to a restart if the reload fails.
- **\*\*kwargs** – additional parameters to pass to the init system when managing services. These will be passed as key=value parameters to the init system's commandline. kwargs are ignored for init systems not allowing additional parameters via the commandline (systemd).

`charmhelpers.core.host.service_restart` (*service\_name*, *\*\*kwargs*)

Restart a system service.

The specified service name is managed via the system level init system. Some init systems (e.g. upstart) require that additional arguments be provided in order to directly control service instances whereas other init systems allow for addressing instances of a service directly by name (e.g. systemd).

The kwargs allow for the additional parameters to be passed to underlying init systems for those systems which require/allow for them. For example, the ceph-osd upstart script requires the `id` parameter to be passed along in order to identify which running daemon should be restarted. The following example restarts the ceph-osd service for instance `id=4`:

```
service_restart('ceph-osd', id=4)
```

### Parameters

- **service\_name** – the name of the service to restart
- **\*\*kwargs** – additional parameters to pass to the init system when managing services. These will be passed as key=value parameters to the init system's commandline. kwargs are ignored for init systems not allowing additional parameters via the commandline (systemd).

`charmhelpers.core.host.service_resume` (*service\_name*, *init\_dir='/etc/init'*, *initd\_dir='/etc/init.d'*, *\*\*kwargs*)

Resume a system service.

Reenable starting again at boot. Start the service.

### Parameters

- **service\_name** – the name of the service to resume
- **init\_dir** – the path to the init dir
- **dir** (*initd*) – the path to the initd dir
- **\*\*kwargs** – additional parameters to pass to the init system when managing services. These will be passed as key=value parameters to the init system's commandline. kwargs are ignored for systemd enabled systems.

`charmhelpers.core.host.service_running` (*service\_name*, *\*\*kwargs*)

Determine whether a system service is running.

#### Parameters

- **service\_name** – the name of the service
- **\*\*kwargs** – additional args to pass to the service command. This is used to pass additional key=value arguments to the service command line for managing specific instance units (e.g. service ceph-osd status id=2). The kwargs are ignored in systemd services.

`charmhelpers.core.host.service_start` (*service\_name*, *\*\*kwargs*)

Start a system service.

The specified service name is managed via the system level init system. Some init systems (e.g. upstart) require that additional arguments be provided in order to directly control service instances whereas other init systems allow for addressing instances of a service directly by name (e.g. systemd).

The kwargs allow for the additional parameters to be passed to underlying init systems for those systems which require/allow for them. For example, the ceph-osd upstart script requires the id parameter to be passed along in order to identify which running daemon should be reloaded. The following example stops the ceph-osd service for instance id=4:

```
service_stop('ceph-osd', id=4)
```

#### Parameters

- **service\_name** – the name of the service to stop
- **\*\*kwargs** – additional parameters to pass to the init system when managing services. These will be passed as key=value parameters to the init system's commandline. kwargs are ignored for systemd enabled systems.

`charmhelpers.core.host.service_stop` (*service\_name*, *\*\*kwargs*)

Stop a system service.

The specified service name is managed via the system level init system. Some init systems (e.g. upstart) require that additional arguments be provided in order to directly control service instances whereas other init systems allow for addressing instances of a service directly by name (e.g. systemd).

The kwargs allow for the additional parameters to be passed to underlying init systems for those systems which require/allow for them. For example, the ceph-osd upstart script requires the id parameter to be passed along in order to identify which running daemon should be reloaded. The following example stops the ceph-osd service for instance id=4:

```
service_stop('ceph-osd', id=4)
```

#### Parameters

- **service\_name** – the name of the service to stop
- **\*\*kwargs** – additional parameters to pass to the init system when managing services. These will be passed as key=value parameters to the init system's commandline. kwargs are ignored for systemd enabled systems.

`charmhelpers.core.host.set_nic_mtu` (*nic*, *mtu*)

Set the Maximum Transmission Unit (MTU) on a network interface.

`charmhelpers.core.host.symlink` (*source*, *destination*)

Create a symbolic link

`charmhelpers.core.host.uid_exists` (*uid*)

Check if a uid exists

`charmhelpers.core.host.umount` (*mountpoint*, *persist=False*)  
Unmount a filesystem

`charmhelpers.core.host.updatedb` (*updatedb\_text*, *new\_path*)

`charmhelpers.core.host.user_exists` (*username*)  
Check if a user exists

`charmhelpers.core.host.write_file` (*path*, *content*, *owner='root'*, *group='root'*, *perms=292*)  
Create or overwrite a file with the contents of a byte string.

### 3.1.5 charmhelpers.core.strutils

**class** `charmhelpers.core.strutils.BasicStringComparator` (*item*)  
Bases: object

Provides a class that will compare strings from an iterator type object. Used to provide > and < comparisons on strings that may not necessarily be alphanumerically ordered. e.g. OpenStack or Ubuntu releases AFTER the z-wrap.

`charmhelpers.core.strutils.bool_from_string` (*value*)  
Interpret string value as boolean.

Returns True if value translates to True otherwise False.

`charmhelpers.core.strutils.bytes_from_string` (*value*)  
Interpret human readable string value as bytes.

Returns int

### 3.1.6 charmhelpers.core.sysctl

`charmhelpers.core.sysctl.create` (*sysctl\_dict*, *sysctl\_file*, *ignore=False*)  
Creates a sysctl.conf file from a YAML associative array

#### Parameters

- **sysctl\_dict** (*str*) – a dict or YAML-formatted string of sysctl options eg “{ ‘kernel.max\_pid’: 1337 }”
- **sysctl\_file** (*str* or *unicode*) – path to the sysctl file to be saved
- **ignore** (*bool*) – If True, ignore “unknown variable” errors.

**Returns** None

### 3.1.7 charmhelpers.core.templating

`charmhelpers.core.templating.render` (*source*, *target*, *context*, *owner='root'*, *group='root'*,  
*perms=292*, *templates\_dir=None*, *encoding='UTF-8'*,  
*template\_loader=None*, *config\_template=None*)

Render a template.

The *source* path, if not absolute, is relative to the *templates\_dir*.

The *target* path should be absolute. It can also be *None*, in which case no file will be written.

The context should be a dict containing the values to be replaced in the template.

*config\_template* may be provided to render from a provided template instead of loading from a file.

The *owner*, *group*, and *perms* options will be passed to *write\_file*.

If omitted, *templates\_dir* defaults to the *templates* folder in the charm.

The rendered template will be written to the file as well as being returned as a string.

Note: Using this requires `python-jinja2` or `python3-jinja2`; if it is not installed, calling this will attempt to use `charmhelpers.fetch.apt_install` to install it.

### 3.1.8 charmhelpers.core.unitdata

#### Intro

A simple way to store state in units. This provides a key value storage with support for versioned, transactional operation, and can calculate deltas from previous values to simplify unit logic when processing changes.

#### Hook Integration

There are several extant frameworks for hook execution, including

- `charmhelpers.core.hookenv.Hooks`
- `charmhelpers.core.services.ServiceManager`

The storage classes are framework agnostic, one simple integration is via the `HookData` contextmanager. It will record the current hook execution environment (including relation data, config data, etc.), setup a transaction and allow easy access to the changes from previously seen values. One consequence of the integration is the reservation of particular keys (`'rels'`, `'unit'`, `'env'`, `'config'`, `'charm_revisions'`) for their respective values.

Here's a fully worked integration example using `hookenv.Hooks`:

```
from charmhelper.core import hookenv, unitdata

hook_data = unitdata.HookData()
db = unitdata.kv()
hooks = hookenv.Hooks()

@hooks.hook
def config_changed():
    # Print all changes to configuration from previously seen
    # values.
    for changed, (prev, cur) in hook_data.conf.items():
        print('config changed', changed,
              'previous value', prev,
              'current value', cur)

    # Get some unit specific bookeeping
    if not db.get('pkg_key'):
        key = urllib.urlopen('https://example.com/pkg_key').read()
        db.set('pkg_key', key)

    # Directly access all charm config as a mapping.
    conf = db.getrange('config', True)

    # Directly access all relation data as a mapping
    rels = db.getrange('rels', True)
```

(continues on next page)

(continued from previous page)

```
if __name__ == '__main__':
    with hook_data():
        hook.execute()
```

A more basic integration is via the `hook_scope` context manager which simply manages transaction scope (and records hook name, and timestamp):

```
>>> from unitdata import kv
>>> db = kv()
>>> with db.hook_scope('install'):
...     # do work, in transactional scope.
...     db.set('x', 1)
>>> db.get('x')
1
```

## Usage

Values are automatically json de/serialized to preserve basic typing and complex data struct capabilities (dicts, lists, ints, booleans, etc).

Individual values can be manipulated via `get/set`:

```
>>> kv.set('y', True)
>>> kv.get('y')
True

# We can set complex values (dicts, lists) as a single key.
>>> kv.set('config', {'a': 1, 'b': True})

# Also supports returning dictionaries as a record which
# provides attribute access.
>>> config = kv.get('config', record=True)
>>> config.b
True
```

Groups of keys can be manipulated with `update/getrange`:

```
>>> kv.update({'z': 1, 'y': 2}, prefix="gui.")
>>> kv.getrange('gui.', strip=True)
{'z': 1, 'y': 2}
```

When updating values, its very helpful to understand which values have actually changed and how have they changed. The storage provides a `delta` method to provide for this:

```
>>> data = {'debug': True, 'option': 2}
>>> delta = kv.delta(data, 'config.')
>>> delta.debug.previous
None
>>> delta.debug.current
True
>>> delta
{'debug': (None, True), 'option': (None, 2)}
```

Note the `delta` method does not persist the actual change, it needs to be explicitly saved via `'update'` method:



```
>>> kv.update(data, 'config.')
```

Values modified in the context of a hook scope retain historical values associated to the hookname.

```
>>> with db.hook_scope('config-changed'):
...     db.set('x', 42)
>>> db.gethistory('x')
[(1, u'x', 1, u'install', u'2015-01-21T16:49:30.038372'),
 (2, u'x', 42, u'config-changed', u'2015-01-21T16:49:30.038786')]
```

**class** charmhelpers.core.unitdata.**Delta** (*previous, current*)  
Bases: tuple

**current**

Alias for field number 1

**previous**

Alias for field number 0

**class** charmhelpers.core.unitdata.**DeltaSet**  
Bases: *charmhelpers.core.unitdata.Record*

**class** charmhelpers.core.unitdata.**HookData**  
Bases: object

Simple integration for existing hook exec frameworks.

Records all unit information, and stores deltas for processing by the hook.

Sample:

```
from charmhelper.core import hookenv, unitdata

changes = unitdata.HookData()
db = unitdata.kv()
hooks = hookenv.Hooks()

@hooks.hook
def config_changed():
    # View all changes to configuration
    for changed, (prev, cur) in changes.conf.items():
        print('config changed', changed,
              'previous value', prev,
              'current value', cur)

    # Get some unit specific bookeeping
    if not db.get('pkg_key'):
        key = urllib.urlopen('https://example.com/pkg_key').read()
        db.set('pkg_key', key)

if __name__ == '__main__':
    with changes():
        hook.execute()
```

**class** charmhelpers.core.unitdata.**Record**  
Bases: dict

**class** charmhelpers.core.unitdata.**Storage** (*path=None*)  
Bases: object

Simple key value database for local unit state within charms.

Modifications are not persisted unless `flush()` is called.

To support dicts, lists, integer, floats, and booleans values are automatically json encoded/decoded.

Note: to facilitate unit testing, `':memory:'` can be passed as the path parameter which causes sqlite3 to only build the db in memory. This should only be used for testing purposes.

**close** ()

**debug** (*fh*=<\_io.TextIOWrapper name='<stderr>' mode='w' encoding='UTF-8'>)

**delta** (*mapping*, *prefix*)

return a delta containing values that have changed.

**flush** (*save*=True)

**get** (*key*, *default*=None, *record*=False)

**gethistory** (*key*, *deserialize*=False)

**getrange** (*key\_prefix*, *strip*=False)

Get a range of keys starting with a common prefix as a mapping of keys to values.

**Parameters**

- **key\_prefix** (*str*) – Common prefix among all keys
- **strip** (*bool*) – Optionally strip the common prefix from the key names in the returned dict

**Return dict** A (possibly empty) dict of key-value mappings

**hook\_scope** (*name*="")

Scope all future interactions to the current hook execution revision.

**set** (*key*, *value*)

Set a value in the database.

**Parameters**

- **key** (*str*) – Key to set the value for
- **value** – Any JSON-serializable value to be set

**unset** (*key*)

Remove a key from the database entirely.

**unsetrange** (*keys*=None, *prefix*="")

Remove a range of keys starting with a common prefix, from the database entirely.

**Parameters**

- **keys** (*list*) – List of keys to remove.
- **prefix** (*str*) – Optional prefix to apply to all keys in *keys* before removing.

**update** (*mapping*, *prefix*="")

Set the values of multiple keys at once.

**Parameters**

- **mapping** (*dict*) – Mapping of keys to values
- **prefix** (*str*) – Optional prefix to apply to all keys in *mapping* before setting

`charmhelpers.core.unitdata.kv()`

### 3.1.9 charmhelpers.core.services

#### charmhelpers.core.services.base

<i>ManagerCallback</i>	Special case of a callback that takes the <i>ServiceManager</i> instance in addition to the service name.
<i>PortManagerCallback</i>	Callback class that will open or close ports, for use as either a start or stop action.
<i>ServiceManager</i>	
<code>close_ports</code>	Callback class that will open or close ports, for use as either a start or stop action.
<code>manage_ports</code>	Callback class that will open or close ports, for use as either a start or stop action.
<code>open_ports</code>	Callback class that will open or close ports, for use as either a start or stop action.
<i>service_restart</i>	Wrapper around <code>host.service_restart</code> to prevent spurious “unknown service” messages in the logs.
<i>service_stop</i>	Wrapper around <code>host.service_stop</code> to prevent spurious “unknown service” messages in the logs.

**class** `charmhelpers.core.services.base.ServiceManager` (*services=None*)

Bases: object

**fire\_event** (*event\_name, service\_name, default=None*)

Fire a `data_ready`, `data_lost`, `start`, or `stop` event on a given service.

**get\_service** (*service\_name*)

Given the name of a registered service, return its service definition.

**is\_ready** (*service\_name*)

Determine if a registered service is ready, by checking its ‘`required_data`’.

A ‘`required_data`’ item can be any mapping type, and is considered ready if `bool(item)` evaluates as `True`.

**manage** ()

Handle the current hook by doing The Right Thing with the registered services.

**provide\_data** ()

Set the relation data for each provider in the `provided_data` list.

A provider must have a `name` attribute, which indicates which relation to set data on, and a `provide_data()` method, which returns a dict of data to set.

The `provide_data()` method can optionally accept two parameters:

- `remote_service` The name of the remote service that the data will be provided to. The `provide_data()` method will be called once for each connected service (not unit). This allows the method to tailor its data to the given service.
- `service_ready` Whether or not the service definition had all of its requirements met, and thus the `data_ready` callbacks run.

Note that the `provided_data` methods are now called **after** the `data_ready` callbacks are run. This gives the `data_ready` callbacks a chance to generate any data necessary for the providing to the remote services.

**reconfigure\_services** (*\*service\_names*)

Update all files for one or more registered services, and, if ready, optionally restart them.

If no service names are given, reconfigures all registered services.

**save\_lost** (*service\_name*)

Save an indicator that the given service is no longer data\_ready.

**save\_ready** (*service\_name*)

Save an indicator that the given service is now data\_ready.

**stop\_services** (*\*service\_names*)

Stop one or more registered services, by name.

If no service names are given, stops all registered services.

**was\_ready** (*service\_name*)

Determine if the given service was previously data\_ready.

**class** charmhelpers.core.services.base.**ManagerCallback**

Bases: object

Special case of a callback that takes the *ServiceManager* instance in addition to the service name.

Subclasses should implement `__call__` which should accept three parameters:

- *manager* The *ServiceManager* instance
- *service\_name* The name of the service it's being triggered for
- *event\_name* The name of the event that this callback is handling

**class** charmhelpers.core.services.base.**PortManagerCallback**

Bases: *charmhelpers.core.services.base.ManagerCallback*

Callback class that will open or close ports, for use as either a start or stop action.

**ports\_contains** (*port, ports*)

charmhelpers.core.services.base.**service\_restart** (*service\_name*)

Wrapper around host.service\_restart to prevent spurious “unknown service” messages in the logs.

charmhelpers.core.services.base.**service\_stop** (*service\_name*)

Wrapper around host.service\_stop to prevent spurious “unknown service” messages in the logs.

## charmhelpers.core.services.helpers

<code>HttpRelation</code>	Relation context for the <i>http</i> interface.
<code>MysqlRelation</code>	Relation context for the <i>mysql</i> interface.
<code>RelationContext</code>	Base class for a context generator that gets relation data from juju.
<code>RequiredConfig</code>	Data context that loads config options with one or more mandatory options.
<code>StoredContext</code>	A data context that always returns the data that it was first created with.
<code>TemplateCallback</code>	Callback class that will render a Jinja2 template, for use as a ready action.
<code>render_template</code>	alias of <i>charmhelpers.core.services.helpers.TemplateCallback</i>
<code>template</code>	alias of <i>charmhelpers.core.services.helpers.TemplateCallback</i>

```
class charmhelpers.core.services.helpers.RelationContext (name=None, additional_required_keys=None)
```

Bases: dict

Base class for a context generator that gets relation data from juju.

Subclasses must provide the attributes *name*, which is the name of the interface of interest, *interface*, which is the type of the interface of interest, and *required\_keys*, which is the set of keys required for the relation to be considered complete. The data for all interfaces matching the *name* attribute that are complete will be used to populate the dictionary values (see *get\_data*, below).

The generated context will be namespaced under the relation *name*, to prevent potential naming conflicts.

#### Parameters

- **name** (*str*) – Override the relation *name*, since it can vary from charm to charm
- **additional\_required\_keys** (*list*) – Extend the list of *required\_keys*

#### get\_data()

Retrieve the relation data for each unit involved in a relation and, if complete, store it in a list under *self[self.name]*. This is automatically called when the RelationContext is instantiated.

The units are sorted lexicographically first by the service ID, then by the unit ID. Thus, if an interface has two other services, 'db:1' and 'db:2', with 'db:1' having two units, 'wordpress/0' and 'wordpress/1', and 'db:2' having one unit, 'mediawiki/0', all of which have a complete set of data, the relation data for the units will be stored in the order: 'wordpress/0', 'wordpress/1', 'mediawiki/0'.

If you only care about a single unit on the relation, you can just access it as `{{ interface[0]['key'] }}`. However, if you can at all support multiple units on a relation, you should iterate over the list, like:

```
{% for unit in interface -%}
  {{ unit['key'] }}{% if not loop.last %},{% endif %}
{%- endfor %}
```

Note that since all sets of relation data from all related services and units are in a single list, if you need to know which service or unit a set of data came from, you'll need to extend this class to preserve that information.

**interface = None**

**is\_ready()**

Returns True if all of the *required\_keys* are available from any units.

**name = None**

**provide\_data()**

Return data to be relation\_set for this interface.

```
class charmhelpers.core.services.helpers.TemplateCallback (source, target,
                                                         owner='root',
                                                         group='root',
                                                         perms=292,
                                                         on_change_action=None,
                                                         tem-
                                                         plate_loader=None)
```

Bases: *charmhelpers.core.services.base.ManagerCallback*

Callback class that will render a Jinja2 template, for use as a ready action.

#### Parameters

- **source** (*str*) – The template source file, relative to *\$CHARM\_DIR/templates*

- **target** (*str*) – The target to write the rendered template to (or None)
- **owner** (*str*) – The owner of the rendered file
- **group** (*str*) – The group of the rendered file
- **perms** (*int*) – The permissions of the rendered file
- **on\_change\_action** (*partial*) – functools partial to be executed when rendered file changes
- **loader** **template\_loader** (*jinja2*) – A jinja2 template loader

**Return str** The rendered template

```
charmhelpers.core.services.helpers.render_template
    alias of charmhelpers.core.services.helpers.TemplateCallback

charmhelpers.core.services.helpers.template
    alias of charmhelpers.core.services.helpers.TemplateCallback
```

## 3.2 charmhelpers.contrib package

### 3.2.1 charmhelpers.contrib.ansible package

Charm Helpers ansible - declare the state of your machines.

This helper enables you to declare your machine state, rather than program it procedurally (and have to test each change to your procedures). Your install hook can be as simple as:

```
{ {{
import charmhelpers.contrib.ansible

def install():
    charmhelpers.contrib.ansible.install_ansible_support()
    charmhelpers.contrib.ansible.apply_playbook('playbooks/install.yaml')
}}}
```

and won't need to change (nor will its tests) when you change the machine state.

All of your juju config and relation-data are available as template variables within your playbooks and templates. An install playbook looks something like:

```
{ {{
---
- hosts: localhost
  user: root

  tasks:
    - name: Add private repositories.
      template:
        src: ../templates/private-repositories.list.jinja2
        dest: /etc/apt/sources.list.d/private.list

    - name: Update the cache.
      apt: update_cache=yes
```

(continues on next page)

(continued from previous page)

```

- name: Install dependencies.
  apt: pkg={{ item }}
  with_items:
    - python-mimeparse
    - python-webob
    - sunburnt

- name: Setup groups.
  group: name={{ item.name }} gid={{ item.gid }}
  with_items:
    - { name: 'deploy_user', gid: 1800 }
    - { name: 'service_user', gid: 1500 }

...
}}}
```

Read more online about [playbooks](#) and standard ansible [modules](#).

A further feature of the ansible hooks is to provide a light weight “action” scripting tool. This is a decorator that you apply to a function, and that function can now receive cli args, and can pass extra args to the playbook.

e.g.

```
@hooks.action() def some_action(amount, force="False"):
```

```
    "Usage: some-action AMOUNT [force=True]" # <- shown on error # process the arguments # do some
    calls # return extra-vars to be passed to ansible-playbook return {
```

```
        'amount': int(amount), 'type': force,
    }
```

You can now create a symlink to hooks.py that can be invoked like a hook, but with cli params:

```
# link actions/some-action to hooks/hooks.py
```

```
actions/some-action amount=10 force=true
```

```
class charmhelpers.contrib.ansible.AnsibleHooks (playbook_path, default_hooks=None)
    Bases: charmhelpers.core.hookenv.Hooks
```

Run a playbook with the hook-name as the tag.

This helper builds on the standard hookenv.Hooks helper, but additionally runs the playbook with the hook-name specified using `-tags` (ie. running all the tasks tagged with the hook-name).

Example:

```
hooks = AnsibleHooks(playbook_path='playbooks/my_machine_state.yaml')

# All the tasks within my_machine_state.yaml tagged with 'install'
# will be run automatically after do_custom_work()
@hooks.hook()
def install():
    do_custom_work()

# For most of your hooks, you won't need to do anything other
# than run the tagged tasks for the hook:
@hooks.hook('config-changed', 'start', 'stop')
def just_use_playbook():
    pass
```

(continues on next page)

(continued from previous page)

```
# As a convenience, you can avoid the above noop function by specifying
# the hooks which are handled by ansible-only and they'll be registered
# for you:
# hooks = AnsibleHooks(
#     'playbooks/my_machine_state.yaml',
#     default_hooks=['config-changed', 'start', 'stop'])

if __name__ == "__main__":
    # execute a hook based on the name the program is called by
    hooks.execute(sys.argv)
```

**action** (\**action\_names*)  
Decorator, registering them as actions

**execute** (*args*)  
Execute the hook followed by the playbook using the hook as tag.

**register\_action** (*name, function*)  
Register a hook

charmhelpers.contrib.ansible.**apply\_playbook** (*playbook, tags=None, extra\_vars=None*)

charmhelpers.contrib.ansible.**install\_ansible\_support** (*from\_ppa=True,*  
*ppa\_location='ppa:rquillo/ansible'*)

Installs the ansible package.

By default it is installed from the [PPA](#) linked from the ansible [website](#) or from a ppa specified by a charm config..

If from\_ppa is empty, you must ensure that the package is available from a configured repository.

### 3.2.2 charmhelpers.contrib.charmhelpers package

charmhelpers.contrib.charmhelpers.**unit\_info** (*service\_name, item\_name, data=None,*  
*unit=None*)

charmhelpers.contrib.charmhelpers.**wait\_for\_machine** (*num\_machines=1, timeout=300*)  
Wait *timeout* seconds for *num\_machines* machines to come up.

This wait\_for... function can be called by other wait\_for functions whose timeouts might be too short in situations where only a bare Juju setup has been bootstrapped.

**Returns** A tuple of (num\_machines, time\_taken). This is used for testing.

charmhelpers.contrib.charmhelpers.**wait\_for\_page\_contents** (*url, contents, time-*  
*out=120, vali-*  
*date=None*)

charmhelpers.contrib.charmhelpers.**wait\_for\_relation** (*service\_name, relation\_name,*  
*timeout=120*)

Wait *timeout* seconds for a given relation to come up.

charmhelpers.contrib.charmhelpers.**wait\_for\_unit** (*service\_name, timeout=480*)  
Wait *timeout* seconds for a given service name to come up.

### 3.2.3 charmhelpers.contrib.charmsupport package



**charmhelpers.contrib.charmsupport.nrpe module**

Compatibility with the nrpe-external-master charm

```
class charmhelpers.contrib.charmsupport.nrpe.Check (shortname, description,
                                                check_cmd)
    Bases: object
    remove (hostname)
    run ()
    service_template = '\n#-----\n# This file'
    shortname_re = '[A-Za-z0-9-_.@]+'
    write (nagios_context, hostname, nagios_servicegroups)
    write_service_config (nagios_context, hostname, nagios_servicegroups)
```

```
exception charmhelpers.contrib.charmsupport.nrpe.CheckException
    Bases: Exception
```

```
class charmhelpers.contrib.charmsupport.nrpe.NRPE (hostname=None, primary=True)
    Bases: object
    add_check (*args, **kwargs)
    homedir = '/var/lib/nagios'
    nagios_exportdir = '/var/lib/nagios/export'
    nagios_logdir = '/var/log/nagios'
    nrpe_confdir = '/etc/nagios/nrpe.d'
    remove_check (*args, **kwargs)
    write ()
```

charmhelpers.contrib.charmsupport.nrpe.**add\_haproxy\_checks** (*nrpe*, *unit\_name*)  
Add checks for each service in list

**Parameters**

- **nrpe** (*NRPE*) – NRPE object to add check to
- **unit\_name** (*str*) – Unit name to use in check description

```
charmhelpers.contrib.charmsupport.nrpe.add_init_service_checks (nrpe, services,
                                                                unit_name,
                                                                immediate_check=True)
```

Add checks for each service in list

**Parameters**

- **nrpe** (*NRPE*) – NRPE object to add check to
- **services** (*list*) – List of services to check
- **unit\_name** (*str*) – Unit name to use in check description
- **immediate\_check** (*bool*) – For sysv init, run the service check immediately

```
charmhelpers.contrib.charmsupport.nrpe.copy_nrpe_checks (nrpe_files_dir=None)  
Copy the nrpe checks into place
```

`charmhelpers.contrib.charmsupport.nrpe.get_nagios_hostcontext` (*relation\_name='nrpe-external-master'*)

Query relation with nrpe subordinate, return the nagios\_host\_context

**Parameters** `relation_name` (*str*) – Name of relation nrpe sub joined to

`charmhelpers.contrib.charmsupport.nrpe.get_nagios_hostname` (*relation\_name='nrpe-external-master'*)

Query relation with nrpe subordinate, return the nagios\_hostname

**Parameters** `relation_name` (*str*) – Name of relation nrpe sub joined to

`charmhelpers.contrib.charmsupport.nrpe.get_nagios_unit_name` (*relation\_name='nrpe-external-master'*)

Return the nagios unit name prepended with host\_context if needed

**Parameters** `relation_name` (*str*) – Name of relation nrpe sub joined to

### charmhelpers.contrib.charmsupport.volumes module

Functions for managing volumes in juju units. One volume is supported per unit. Subordinates may have their own storage, provided it is on its own partition.

Configuration stanzas:

```

volume-ephemeral:
  type: boolean
  default: true
  description: >
    If false, a volume is mounted as sepecified in "volume-map"
    If true, ephemeral storage will be used, meaning that log data
      will only exist as long as the machine. YOU HAVE BEEN WARNED.
volume-map:
  type: string
  default: {}
  description: >
    YAML map of units to device names, e.g:
    "{ rsyslog/0: /dev/vdb, rsyslog/1: /dev/vdb }"
    Service units will raise a configure-error if volume-ephemeral
is 'true' and no volume-map value is set. Use 'juju set' to set a
    value and 'juju resolved' to complete configuration.

```

Usage:

```

from charmsupport.volumes import configure_volume, VolumeConfigurationError
from charmsupport.hookenv import log, ERROR
def post_mount_hook():
    stop_service('myservice')
def post_mount_hook():
    start_service('myservice')

if __name__ == '__main__':
    try:
        configure_volume(before_change=pre_mount_hook,
                        after_change=post_mount_hook)
    except VolumeConfigurationError:
        log('Storage could not be configured', ERROR)

```

**exception** `charmhelpers.contrib.charmsupport.volumes.VolumeConfigurationError`  
 Bases: Exception

Volume configuration data is missing or invalid

```
charmhelpers.contrib.charmsupport.volumes.configure_volume (before_change=<function
                                                    <lambda>>, af-
                                                    ter_change=<function
                                                    <lambda>>)
```

Set up storage (or don't) according to the charm's volume configuration. Returns the mount point or "ephemeral". `before_change` and `after_change` are optional functions to be called if the volume configuration changes.

```
charmhelpers.contrib.charmsupport.volumes.get_config()
```

Gather and sanity-check volume configuration data

```
charmhelpers.contrib.charmsupport.volumes.managed_mounts()
```

List of all mounted managed volumes

```
charmhelpers.contrib.charmsupport.volumes.mount_volume (config)
```

```
charmhelpers.contrib.charmsupport.volumes.unmount_volume (config)
```

### 3.2.4 charmhelpers.contrib.hahelpers package

#### charmhelpers.contrib.hahelpers.apache module

```
charmhelpers.contrib.hahelpers.apache.get_ca_cert()
```

```
charmhelpers.contrib.hahelpers.apache.get_cert (cn=None)
```

```
charmhelpers.contrib.hahelpers.apache.install_ca_cert (ca_cert)
```

```
charmhelpers.contrib.hahelpers.apache.retrieve_ca_cert (cert_file)
```

#### charmhelpers.contrib.hahelpers.cluster module

Helpers for clustering and determining "cluster leadership" and other clustering-related helpers.

```
exception charmhelpers.contrib.hahelpers.cluster.CRMDCNotFound
```

Bases: Exception

```
exception charmhelpers.contrib.hahelpers.cluster.CRMResourceNotFound
```

Bases: Exception

```
exception charmhelpers.contrib.hahelpers.cluster.HAIncompleteConfig
```

Bases: Exception

```
exception charmhelpers.contrib.hahelpers.cluster.HAIncorrectConfig
```

Bases: Exception

```
charmhelpers.contrib.hahelpers.cluster.canonical_url (configs, vip_setting='vip')
```

Returns the correct HTTP URL to this host given the state of HTTPS configuration and hacluster.

**:configs** [OSTemplateRenderer: A config templating object to inspect for] a complete https context.

**Vip\_setting** str: Setting in charm config that specifies VIP address.

```
charmhelpers.contrib.hahelpers.cluster.determine_apache_port (public_port,
                                                    singlen-
                                                    ode_mode=False)
```

Description: Determine correct apache listening port based on public IP + state of the cluster.

public\_port: int: standard public port for given service

singlenode\_mode: boolean: Shuffle ports when only a single unit is present

returns: int: the correct listening port for the HAProxy service

`charmhelpers.contrib.hahelpers.cluster.determine_api_port` (*public\_port*, *singlenode\_mode=False*)

Determine correct API server listening port based on existence of HTTPS reverse proxy and/or haproxy.

public\_port: int: standard public port for given service

singlenode\_mode: boolean: Shuffle ports when only a single unit is present

returns: int: the correct listening port for the API service

`charmhelpers.contrib.hahelpers.cluster.distributed_wait` (*modulo=None*,  
*wait=None*, *operation\_name='operation'*)

Distribute operations by waiting based on modulo\_distribution

If modulo and or wait are not set, check `config_get` for those values. If config values are not set, default to `modulo=3` and `wait=30`.

**Parameters**

- **modulo** – int The modulo number creates the group distribution
- **wait** – int The constant time wait value
- **operation\_name** – string Operation name for status message i.e. ‘restart’

**Side effect** Calls `config_get()`

**Side effect** Calls `log()`

**Side effect** Calls `status_set()`

**Side effect** Calls `time.sleep()`

`charmhelpers.contrib.hahelpers.cluster.eligible_leader` (*resource*)

`charmhelpers.contrib.hahelpers.cluster.get_hacluster_config` (*exclude\_keys=None*)

Obtains all relevant configuration from charm configuration required for initiating a relation to hacluster:

ha-bindiface, ha-mcastport, vip, os-internal-hostname, os-admin-hostname, os-public-hostname, os-access-hostname

param: `exclude_keys`: list of setting key(s) to be excluded. returns: dict: A dict containing settings keyed by setting name. raises: `HAIncompleteConfig` if settings are missing or incorrect.

`charmhelpers.contrib.hahelpers.cluster.https` ()

Determines whether enough data has been provided in configuration or relation data to configure HTTPS .

returns: boolean

`charmhelpers.contrib.hahelpers.cluster.is_clustered` ()

`charmhelpers.contrib.hahelpers.cluster.is_crm_dc` ()

Determine leadership by querying the pacemaker Designated Controller

`charmhelpers.contrib.hahelpers.cluster.is_elected_leader` (*resource*)

Returns True if the charm executing this is the elected cluster leader.

**It relies on two mechanisms to determine leadership:** 1. If juju is sufficiently new and leadership election is supported, the `is_leader` command will be used. 2. If the charm is part of a corosync cluster, call `corosync` to determine leadership. 3. If the charm is not part of a corosync cluster, the leader is determined as being “the alive unit with the lowest unit numer”. In other words, the oldest surviving unit.

`charmhelpers.contrib.hahelpers.cluster.is_leader` (*resource*)

`charmhelpers.contrib.hahelpers.cluster.oldest_peer` (*peers*)

Determines who the oldest peer is by comparing unit numbers.

`charmhelpers.contrib.hahelpers.cluster.peer_ips` (*peer\_relation='cluster',  
addr\_key='private-address'*)

Return a dict of peers and their private-address

`charmhelpers.contrib.hahelpers.cluster.peer_units` (*peer\_relation='cluster'*)

`charmhelpers.contrib.hahelpers.cluster.valid_hacluster_config` ()

Check that either vip or dns-ha is set. If dns-ha then one of os-*\**-hostname must be set.

Note: ha-bindiface and ha-macastport both have defaults and will always be set. We only care that either vip or dns-ha is set.

**Returns** boolean: valid config returns true.

raises: HAIncompatibleConfig if settings conflict. raises: HAIncompleteConfig if settings are missing.

### 3.2.5 charmhelpers.contrib.network package

#### charmhelpers.contrib.network.ovs package

Helpers for interacting with OpenvSwitch

`charmhelpers.contrib.network.ovs.add_bridge` (*name, datapath\_type=None*)

Add the named bridge to openvswitch

`charmhelpers.contrib.network.ovs.add_bridge_port` (*name, port, promisc=False*)

Add a port to the named openvswitch bridge

`charmhelpers.contrib.network.ovs.add_ovsbridge_linuxbridge` (*name, bridge*)

Add linux bridge to the named openvswitch bridge :param name: Name of ovs bridge to be added to Linux bridge :param bridge: Name of Linux bridge to be added to ovs bridge :returns: True if veth is added between ovs bridge and linux bridge, False otherwise

`charmhelpers.contrib.network.ovs.check_for_eni_source` ()

Juju removes the source line when setting up interfaces, replace if missing

`charmhelpers.contrib.network.ovs.del_bridge` (*name*)

Delete the named bridge from openvswitch

`charmhelpers.contrib.network.ovs.del_bridge_port` (*name, port*)

Delete a port from the named openvswitch bridge

`charmhelpers.contrib.network.ovs.disable_ipfix` (*bridge*)

Disable IPfix on target bridge. :param bridge: Bridge to modify

`charmhelpers.contrib.network.ovs.enable_ipfix` (*bridge, target*)

Enable IPfix on bridge to target. :param bridge: Bridge to monitor :param target: IPfix remote endpoint

`charmhelpers.contrib.network.ovs.full_restart` ()

Full restart and reload of openvswitch

`charmhelpers.contrib.network.ovs.get_certificate` ()

Read openvswitch certificate from disk

`charmhelpers.contrib.network.ovs.is_linuxbridge_interface` (*port*)

Check if the interface is a linuxbridge bridge :param port: Name of an interface to check whether it is a Linux bridge :returns: True if port is a Linux bridge

`charmhelpers.contrib.network.ovs.port_to_br(port)`

Determine the bridge that contains a port :param port: Name of port to check for :returns str: OVS bridge containing port or None if not found

`charmhelpers.contrib.network.ovs.set_Open_vSwitch_column_value(column_value)`

Calls ovs-vsctl and sets the 'column\_value' in the Open\_vSwitch table.

**Parameters** `column_value` – See <http://www.openvswitch.org/ovs-vsswitchd.conf.db.5.pdf> for details of the relevant values.

:type str :raises CalledProcessException: possibly ovsdb-server is not running

`charmhelpers.contrib.network.ovs.set_manager(manager)`

Set the controller for the local openvswitch

### charmhelpers.contrib.network.ip module

`charmhelpers.contrib.network.ip.assert_charm_supports_ipv6()`

Check whether we are able to support charms ipv6.

`charmhelpers.contrib.network.ip.format_ipv6_addr(address)`

If address is IPv6, wrap it in '[]' otherwise return None.

This is required by most configuration files when specifying IPv6 addresses.

`charmhelpers.contrib.network.ip.get_address_in_network(network, fallback=None, fatal=False)`

Get an IPv4 or IPv6 address within the network from the host.

#### Parameters

- **(str)** (*fallback*) – CIDR presentation format. For example, '192.168.1.0/24'. Supports multiple networks as a space-delimited list.
- **(str)** – If no address is found, return fallback.
- **(boolean)** (*fatal*) – If no address is found, fallback is not set and fatal is True then exit(1).

`charmhelpers.contrib.network.ip.get_bridge_nics(bridge, vnic_dir='/sys/devices/virtual/net')`

Return a list of nics comprising a given bridge on the system.

`charmhelpers.contrib.network.ip.get_bridges(vnic_dir='/sys/devices/virtual/net')`

Return a list of bridges on the system.

`charmhelpers.contrib.network.ip.get_host_ip(hostname, fallback=None)`

Resolves the IP for a given hostname, or returns the input if it is already an IP.

`charmhelpers.contrib.network.ip.get_hostname(address, fqdn=True)`

Resolves hostname for given IP, or returns the input if it is already a hostname.

`charmhelpers.contrib.network.ip.get_iface_addr(iface='eth0', inet_type='AF_INET', inc_aliases=False, fatal=True, exc_list=None)`

Return the assigned IP address for a given interface, if any.

#### Parameters

- **iface** – network interface on which address(es) are expected to be found.
- **inet\_type** – inet address family
- **inc\_aliases** – include alias interfaces in search

- **fatal** – if True, raise exception if address not found
- **exc\_list** – list of addresses to ignore

**Returns** list of ip addresses

`charmhelpers.contrib.network.ip.get_iface_for_address(address, *, key='iface')`

Retrieve an attribute of or the physical interface that the IP address provided could be bound to.

**Parameters**

- **(str)** (*address*) – An individual IPv4 or IPv6 address without a net mask or subnet prefix. For example, '192.168.1.1'.
- **key** – 'iface' for the physical interface name or an attribute of the configured interface, for example 'netmask'.

**Returns str** Requested attribute or None if address is not bindable.

`charmhelpers.contrib.network.ip.get_iface_from_addr(addr)`

Work out on which interface the provided address is configured.

`charmhelpers.contrib.network.ip.get_ipv4_addr(iface='eth0', *, inet_type='AF_INET',  
inc_aliases=False, fatal=True,  
exc_list=None)`

Return the assigned IP address for a given interface, if any.

**Parameters**

- **iface** – network interface on which address(es) are expected to be found.
- **inet\_type** – inet address family
- **inc\_aliases** – include alias interfaces in search
- **fatal** – if True, raise exception if address not found
- **exc\_list** – list of addresses to ignore

**Returns** list of ip addresses

`charmhelpers.contrib.network.ip.get_ipv6_addr(*args, **kwargs)`

`charmhelpers.contrib.network.ip.get_netmask_for_address(address, *,  
key='netmask')`

Retrieve an attribute of or the physical interface that the IP address provided could be bound to.

**Parameters**

- **(str)** (*address*) – An individual IPv4 or IPv6 address without a net mask or subnet prefix. For example, '192.168.1.1'.
- **key** – 'iface' for the physical interface name or an attribute of the configured interface, for example 'netmask'.

**Returns str** Requested attribute or None if address is not bindable.

`charmhelpers.contrib.network.ip.get_relation_ip(interface, cidr_network=None)`

Return this unit's IP for the given interface.

Allow for an arbitrary interface to use with network-get to select an IP. Handle all address selection options including passed cidr network and IPv6.

Usage: `get_relation_ip('amqp', cidr_network='10.0.0/8')`

@param interface: string name of the relation. @param cidr\_network: string CIDR Network to select an address from. @raises Exception if prefer-ipv6 is configured but IPv6 unsupported. @returns IPv6 or IPv4 address

`charmhelpers.contrib.network.ip.is_address_in_network` (*network*, *address*)

Determine whether the provided address is within a network range.

**Parameters**

- **(str)** (*network*) – CIDR presentation format. For example, ‘192.168.1.0/24’.
- **address** – An individual IPv4 or IPv6 address without a net mask or subnet prefix. For example, ‘192.168.1.1’.

**Returns boolean** Flag indicating whether address is in network.

`charmhelpers.contrib.network.ip.is_bridge_member` (*nic*)

Check if a given nic is a member of a bridge.

`charmhelpers.contrib.network.ip.is_ip` (*address*)

Returns True if address is a valid IP address.

`charmhelpers.contrib.network.ip.is_ipv6` (*address*)

Determine whether provided address is IPv6 or not.

`charmhelpers.contrib.network.ip.is_ipv6_disabled` ()

`charmhelpers.contrib.network.ip.no_ip_found_error_out` (*network*)

`charmhelpers.contrib.network.ip.ns_query` (*address*)

`charmhelpers.contrib.network.ip.port_has_listener` (*address*, *port*)

Returns True if the address:port is open and being listened to, else False.

@param address: an IP address or hostname @param port: integer port

Note calls ‘zc’ via a subprocess shell

`charmhelpers.contrib.network.ip.resolve_network_cidr` (*ip\_address*)

Resolves the full address cidr of an ip\_address based on configured network interfaces

`charmhelpers.contrib.network.ip.sniff_iface` (*f*)

Ensure decorated function is called with a value for iface.

If no iface provided, inject net iface inferred from unit private address.

## 3.2.6 charmhelpers.contrib.openstack package

### charmhelpers.contrib.openstack.templates package

### charmhelpers.contrib.openstack.alternatives module

Helper for managing alternatives for file conflict resolution

`charmhelpers.contrib.openstack.alternatives.install_alternative` (*name*, *target*, *source*, *priority=50*)

Install alternative configuration

`charmhelpers.contrib.openstack.alternatives.remove_alternative` (*name*, *source*)

Remove an installed alternative configuration file

**Parameters**

- **name** – string name of the alternative to remove
- **source** – string full path to alternative to remove



**charmhelpers.contrib.openstack.context module**

**class** charmhelpers.contrib.openstack.context.**AMQPContext** (*ssl\_dir=None*,  
*rel\_name='amqp'*,  
*relation\_prefix=None*,  
*relation\_id=None*)

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

**class** charmhelpers.contrib.openstack.context.**ApacheSSLContext**

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

Generates a context for an apache vhost configuration that configures HTTPS reverse proxying for one or many endpoints. Generated context looks something like:

```
{
  'namespace': 'cinder',
  'private_address': 'iscsi.mycinderhost.com',
  'endpoints': [(8776, 8766), (8777, 8767)]
}
```

The endpoints list consists of a tuples mapping external ports to internal ports.

**canonical\_names** ()

Figure out which canonical names clients will access this service.

**configure\_ca** ()

**configure\_cert** (*cn=None*)

**enable\_modules** ()

**external\_ports** = []

**get\_network\_addresses** ()

**For each network configured, return corresponding address and** *hostname* or *vip* (if available).

Returns a list of tuples of the form:

**[(address\_in\_net\_a, hostname\_in\_net\_a), (address\_in\_net\_b, hostname\_in\_net\_b), ...]**

or, if no hostnames(s) available:

**[(address\_in\_net\_a, vip\_in\_net\_a), (address\_in\_net\_b, vip\_in\_net\_b), ...]**

or, if no vip(s) available:

**[(address\_in\_net\_a, address\_in\_net\_a), (address\_in\_net\_b, address\_in\_net\_b), ...]**

**group** = 'root'

**interfaces** = ['https']

**service\_namespace** = None

**user** = 'root'

**class** charmhelpers.contrib.openstack.context.**AppArmorContext** (*profile\_name=None*)

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

Base class for apparmor contexts.

**ctxt**

**install\_aa\_utils** ()

Install packages required for apparmor configuration.

**manually\_disable\_aa\_profile()**  
Manually disable an apparmor profile.

If aa-profile-mode is set to disabled (default) this is required as the template has been written but apparmor is yet unaware of the profile and aa-disable aa-profile fails. Without this the profile would kick into enforce mode on the next service restart.

**setup\_aa\_profile()**  
Setup an apparmor profile. The ctxt dictionary will contain the apparmor profile mode and the apparmor profile name. Makes calls out to aa-disable, aa-complain, or aa-enforce to setup the apparmor profile.

**class** charmhelpers.contrib.openstack.context.**BindHostContext**  
Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

**class** charmhelpers.contrib.openstack.context.**CephContext**  
Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

Generates context for /etc/ceph/ceph.conf templates.

**interfaces** = ['ceph']

**class** charmhelpers.contrib.openstack.context.**DataPortContext**  
Bases: *charmhelpers.contrib.openstack.context.NeutronPortContext*

**class** charmhelpers.contrib.openstack.context.**EnsureDirContext** (*dirname,*  
*\*\*kwargs*)  
Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

Serves as a generic context to create a directory as a side-effect.

Useful for software that supports drop-in files (.d) in conjunction with config option-based templates. Examples include:

- OpenStack oslo.policy drop-in files;
- systemd drop-in config files;
- other software that supports overriding defaults with .d files

Another use-case is when a subordinate generates a configuration for primary to render in a separate directory.

Some software requires a user to create a target directory to be scanned for drop-in files with a specific format. This is why this context is needed to do that before rendering a template.

**class** charmhelpers.contrib.openstack.context.**ExternalPortContext**  
Bases: *charmhelpers.contrib.openstack.context.NeutronPortContext*

**class** charmhelpers.contrib.openstack.context.**HAProxyContext** (*singlenode\_mode=False,*  
*ad-*  
*dress\_types=['admin',*  
*'internal', 'public']*)  
Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

Provides half a context for the haproxy template, which describes all peers to be included in the cluster. Each charm needs to include its own context generator that describes the port mapping.

**Side effect** mkdir is called on HAPROXY\_RUN\_DIR

**interfaces** = ['cluster']

**class** charmhelpers.contrib.openstack.context.**IdentityCredentialsContext** (*service=None,*  
*ser-*  
*vice\_user=None,*  
*rel\_name='identity-*  
*credentials'*)

Bases: *charmhelpers.contrib.openstack.context.IdentityServiceContext*

Context for identity-credentials interface type

```
class charmhelpers.contrib.openstack.context.IdentityServiceContext (service=None,
                                                                    ser-
                                                                    vice_user=None,
                                                                    rel_name='identity-
                                                                    service')
```

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

```
class charmhelpers.contrib.openstack.context.ImageServiceContext
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator

  interfaces = ['image-service']
```

```
class charmhelpers.contrib.openstack.context.InternalEndpointContext
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

Internal endpoint context.

This context provides the endpoint type used for communication between services e.g. between Nova and Cinder internally. Openstack uses Public endpoints by default so this allows admins to optionally use internal endpoints.

```
class charmhelpers.contrib.openstack.context.LibvirtConfigFlagsContext
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

This context provides support for extending the libvirt section through user-defined flags.

```
class charmhelpers.contrib.openstack.context.LogLevelContext
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
class charmhelpers.contrib.openstack.context.LogrotateContext (location, interval,
                                                                    count)
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

Common context generator for logrotate.

```
class charmhelpers.contrib.openstack.context.MemcacheContext (package=None)
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

Memcache context

This context provides options for configuring a local memcache client and server for both IPv4 and IPv6

```
class charmhelpers.contrib.openstack.context.NetworkServiceContext (rel_name='quantum-
                                                                    network-
                                                                    service')
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
class charmhelpers.contrib.openstack.context.NeutronAPIContext
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

Inspects current neutron-plugin-api relation for neutron settings. Return defaults if it is not present.

```
get_neutron_options (rdata)
```

```
interfaces = ['neutron-plugin-api']
```

```
class charmhelpers.contrib.openstack.context.NeutronContext
  Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
calico_ctxt ()
```

```
interfaces = []
```

```
midonet_ctxt ()
nlkv_ctxt ()
network_manager
neutron_ctxt ()
neutron_security_groups
nuage_ctxt ()
nvp_ctxt ()
ovs_ctxt ()
packages
pg_ctxt ()
plugin
```

```
class charmhelpers.contrib.openstack.context.NeutronPortContext
    Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
resolve_ports (ports)
    Resolve NICs not yet bound to bridge(s)

    If hwaddress provided then returns resolved hwaddress otherwise NIC.
```

```
class charmhelpers.contrib.openstack.context.NotificationDriverContext (zmq_relation='zeromq-configuration',
                                                                    amqp_relation='amqp')
    Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
class charmhelpers.contrib.openstack.context.OSConfigFlagContext (charm_flag='config-flags',
                                                                    template_flag='user_config_flags')
    Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

Provides support for user-defined config flags.

Users can define a comma-separated list of key=value pairs in the charm configuration and apply them at any point in any file by using a template flag.

Sometimes users might want config flags inserted within a specific section so this class allows users to specify the template flag name, allowing for multiple template flags (sections) within the same context.

**NOTE: the value of config-flags may be a comma-separated list of** key=value pairs and some Openstack config files support comma-separated lists as values.

```
class charmhelpers.contrib.openstack.context.OSContextGenerator
    Bases: object
```

Base class for all context generators.

```
complete = False
```

```
context_complete (ctxt)
    Check for missing data for the required context data. Set self.missing_data if it exists and return False. Set self.complete if no missing data and return True.
```

```
get_related ()
    Check if any of the context interfaces have relation ids. Set self.related and return True if one of the interfaces has relation ids.
```

```
interfaces = []
```

```
missing_data = []
```

```
related = False
```

```
class charmhelpers.contrib.openstack.context.PhyNICMTUContext
```

```
Bases: charmhelpers.contrib.openstack.context.DataPortContext
```

```
class charmhelpers.contrib.openstack.context.PostgresqlDBContext (database=None)
```

```
Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
interfaces = ['pgsql-db']
```

```
class charmhelpers.contrib.openstack.context.SharedDBContext (database=None,
                                                             user=None, relation_prefix=None,
                                                             ssl_dir=None, relation_id=None)
Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

```
interfaces = ['shared-db']
```

```
class charmhelpers.contrib.openstack.context.SubordinateConfigContext (service,
                                                                        con-
                                                                        fig_file,
                                                                        inter-
                                                                        face)
Bases: charmhelpers.contrib.openstack.context.OSContextGenerator
```

Responsible for inspecting relations to subordinates that may be exporting required config via a json blob.

The subordinate interface allows subordinates to export their configuration requirements to the principle for multiple config files and multiple services. Ie, a subordinate that has interfaces to both glance and nova may export to following yaml blob as json:

```
glance:
  /etc/glance/glance-api.conf:
    sections:
      DEFAULT:
        - [key1, value1]
  /etc/glance/glance-registry.conf:
    MYSECTION:
      - [key2, value2]
nova:
  /etc/nova/nova.conf:
    sections:
      DEFAULT:
        - [key3, value3]
```

It is then up to the principle charms to subscribe this context to the service+config file it is interestd in. Configuration data will be available in the template context, in glance's case, as:

```
ctxt = {
  ... other context ...
  'subordinate_configuration': {
    'DEFAULT': {
      'key1': 'value1',
    },
    'MYSECTION': {
      'key2': 'value2',
    },
  },
}
```

(continues on next page)

(continued from previous page)

```

    }
}

```

**class** charmhelpers.contrib.openstack.context.**SysctlContext**

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

This context check if the 'sysctl' option exists on configuration then creates a file with the loaded contents

**class** charmhelpers.contrib.openstack.context.**SyslogContext**

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

**class** charmhelpers.contrib.openstack.context.**VersionsContext** (*pkg='python-keystone'*)

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

Context to return the openstack and operating system versions.

**class** charmhelpers.contrib.openstack.context.**VolumeAPIContext** (*pkg*)

Bases: *charmhelpers.contrib.openstack.context.InternalEndpointContext*

Volume API context.

This context provides information regarding the volume endpoint to use when communicating between services. It determines which version of the API is appropriate for use.

This value will be determined in the resulting context dictionary returned from calling the VolumeAPIContext object. Information provided by this context is as follows:

**volume\_api\_version:** the volume api version to use, currently 'v2' or 'v3'

**volume\_catalog\_info:** the information to use for a cinder client configuration that consumes API endpoints from the keystone catalog. This is defined as the type:name:endpoint\_type string.

**ctxt**

**class** charmhelpers.contrib.openstack.context.**WSGIWorkerConfigContext** (*name=None, script=None, ad- min\_script=None, pub- lic\_script=None, user=None, group=None, pro- cess\_weight=1.0, ad- min\_process\_weight=0.25, pub- lic\_process\_weight=0.75*)

Bases: *charmhelpers.contrib.openstack.context.WorkerConfigContext*

**class** charmhelpers.contrib.openstack.context.**WorkerConfigContext**

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

**class** charmhelpers.contrib.openstack.context.**ZeroMQContext**

Bases: *charmhelpers.contrib.openstack.context.OSContextGenerator*

**interfaces** = ['zeromq-configuration']

charmhelpers.contrib.openstack.context.**context\_complete** (*ctxt*)

charmhelpers.contrib.openstack.context.**db\_ssl** (*rdata, ctxt, ssl\_dir*)

`charmhelpers.contrib.openstack.context.ensure_packages` (*packages*)

Install but do not upgrade required plugin packages.

`charmhelpers.contrib.openstack.context.get_ipv4_addr` (*iface*='eth0', \*,  
*inet\_type*='AF\_INET',  
*inc\_aliases*=False, *fatal*=True,  
*exc\_list*=None)

Return the assigned IP address for a given interface, if any.

#### Parameters

- **iface** – network interface on which address(es) are expected to be found.
- **inet\_type** – inet address family
- **inc\_aliases** – include alias interfaces in search
- **fatal** – if True, raise exception if address not found
- **exc\_list** – list of addresses to ignore

**Returns** list of ip addresses

`charmhelpers.contrib.openstack.context.get_netmask_for_address` (*address*, \*,  
*key*='netmask')

Retrieve an attribute of or the physical interface that the IP address provided could be bound to.

#### Parameters

- **(str)** (*address*) – An individual IPv4 or IPv6 address without a net mask or subnet prefix. For example, '192.168.1.1'.
- **key** – 'iface' for the physical interface name or an attribute of the configured interface, for example 'netmask'.

**Returns** **str** Requested attribute or None if address is not bindable.

### charmhelpers.contrib.openstack.neutron module

`charmhelpers.contrib.openstack.neutron.determine_dkms_package` ()

Determine which DKMS package should be used based on kernel version

`charmhelpers.contrib.openstack.neutron.headers_package` ()

Ensures correct linux-headers for running kernel are installed, for building DKMS package

`charmhelpers.contrib.openstack.neutron.kernel_version` ()

Retrieve the current major kernel version as a tuple e.g. (3, 13)

`charmhelpers.contrib.openstack.neutron.network_manager` ()

Deals with the renaming of Quantum to Neutron in H and any situations that require compatability (eg, deploying H with network-manager=quantum, upgrading from G).

`charmhelpers.contrib.openstack.neutron.neutron_plugin_attribute` (*plugin*, *attr*,  
*net\_manager*=None)

`charmhelpers.contrib.openstack.neutron.neutron_plugins` ()

`charmhelpers.contrib.openstack.neutron.parse_bridge_mappings` (*mappings*)

Parse bridge mappings.

Mappings must be a space-delimited list of provider:bridge mappings.

Returns dict of the form {provider:bridge}.

`charmhelpers.contrib.openstack.neutron.parse_data_port_mappings` (*mappings*,  
*default\_bridge='br-data'*)

Parse data port mappings.

Mappings must be a space-delimited list of bridge:port.

Returns dict of the form {port:bridge} where ports may be mac addresses or interface names.

`charmhelpers.contrib.openstack.neutron.parse_mappings` (*mappings*,  
*key\_rvalue=False*)

By default mappings are lvalue keyed.

If *key\_rvalue* is True, the mapping will be reversed to allow multiple configs for the same lvalue.

`charmhelpers.contrib.openstack.neutron.parse_vlan_range_mappings` (*mappings*)

Parse vlan range mappings.

Mappings must be a space-delimited list of provider:start:end mappings.

The start:end range is optional and may be omitted.

Returns dict of the form {provider: (start, end)}.

`charmhelpers.contrib.openstack.neutron.quantum_plugins` ()

## charmhelpers.contrib.openstack.templating module

**exception** `charmhelpers.contrib.openstack.templating.OSConfigException`

Bases: Exception

**class** `charmhelpers.contrib.openstack.templating.OSConfigRenderer` (*templates\_dir*,  
*openstack\_release*)

Bases: object

This class provides a common templating system to be used by OpenStack charms. It is intended to help charms share common code and templates, and ease the burden of managing config templates across multiple OpenStack releases.

Basic usage:

```
# import some common context generates from charmhelpers
from charmhelpers.contrib.openstack import context

# Create a renderer object for a specific OS release.
configs = OSConfigRenderer(templates_dir='/tmp/templates',
                           openstack_release='folsom')
# register some config files with context generators.
configs.register(config_file='/etc/nova/nova.conf',
                 contexts=[context.SharedDBContext(),
                           context.AMQPContext()])
configs.register(config_file='/etc/nova/api-paste.ini',
                 contexts=[context.IdentityServiceContext()])
configs.register(config_file='/etc/haproxy/haproxy.conf',
                 contexts=[context.HAProxyContext()])
configs.register(config_file='/etc/keystone/policy.d/extra.cfg',
                 contexts=[context.ExtraPolicyContext(),
                           context.KeystoneContext()],
                 config_template=hookenv.config('extra-policy'))
# write out a single config
```

(continues on next page)



(continued from previous page)

```
configs.write('/etc/nova/nova.conf')
# write out all registered configs
configs.write_all()
```

### OpenStack Releases and template loading

When the object is instantiated, it is associated with a specific OS release. This dictates how the template loader will be constructed.

The constructed loader attempts to load the template from several places in the following order: - from the most recent OS release-specific template dir (if one exists) - the base templates\_dir - a template directory shipped in the charm with this helper file.

For the example above, '/tmp/templates' contains the following structure:

```
/tmp/templates/nova.conf
/tmp/templates/api-paste.ini
/tmp/templates/grizzly/api-paste.ini
/tmp/templates/havana/api-paste.ini
```

Since it was registered with the grizzly release, it first searches the grizzly directory for nova.conf, then the templates dir.

When writing api-paste.ini, it will find the template in the grizzly directory.

If the object were created with folsom, it would fall back to the base templates dir for its api-paste.ini template.

This system should help manage changes in config files through openstack releases, allowing charms to fall back to the most recently updated config template for a given release

The haproxy.conf, since it is not shipped in the templates dir, will be loaded from the module directory's template directory, eg \$CHARM/hooks/charmhelpers/contrib/openstack/templates. This allows us to ship common templates (haproxy, apache) with the helpers.

### Context generators

Context generators are used to generate template contexts during hook execution. Doing so may require inspecting service relations, charm config, etc. When registered, a config file is associated with a list of generators. When a template is rendered and written, all context generates are called in a chain to generate the context dictionary passed to the jinja2 template. See context.py for more info.

#### **complete\_contexts** ()

Returns a list of context interfaces that yield a complete context.

#### **get\_incomplete\_context\_data** (*interfaces*)

Return dictionary of relation status of interfaces and any missing required context data. Example:

```
{'amqp': {'missing_data': ['rabbitmq_password'], 'related': True}, 'zeromq-
configuration': {'related': False}}
```

#### **register** (*config\_file*, *contexts*, *config\_template=None*)

Register a config file with a list of context generators to be called during rendering. config\_template can be used to load a template from a string instead of using template loaders and template files. :param config\_file (str): a path where a config file will be rendered :param contexts (list): a list of context dictionaries with kv pairs :param config\_template (str): an optional template string to use

#### **render** (*config\_file*)

#### **set\_release** (*openstack\_release*)

Resets the template environment and generates a new template loader based on a the new openstack release.

**write** (*config\_file*)

Write a single config file, raises if config file is not registered.

**write\_all** ()

Write out all registered config files.

**class** charmhelpers.contrib.openstack.templating.**OSConfigTemplate** (*config\_file*,  
*contexts*,  
*con-*  
*fig\_template=None*)

Bases: object

Associates a config file template with a list of context generators. Responsible for constructing a template context based on those generators.

**complete\_contexts** ()

Return a list of interfaces that have satisfied contexts.

**context** ()

**is\_string\_template**

**Returns** Boolean if this instance is a template initialised with a string

charmhelpers.contrib.openstack.templating.**get\_loader** (*templates\_dir*, *os\_release*)

Create a jinja2.ChoiceLoader containing template dirs up to and including *os\_release*. If directory template directory is missing at *templates\_dir*, it will be omitted from the loader. *templates\_dir* is added to the bottom of the search list as a base loading dir.

A charm may also ship a templates dir with this module and it will be appended to the bottom of the search list, eg:

```
hooks/charmhelpers/contrib/openstack/templates
```

#### Parameters

- (**str**) (*os\_release*) – Base template directory containing release sub-directories.
- (**str**) – OpenStack release codename to construct template loader.

**Returns** jinja2.ChoiceLoader constructed with a list of jinja2.FilesystemLoaders, ordered in descending order by OpenStack release.

### charmhelpers.contrib.openstack.utils module

**class** charmhelpers.contrib.openstack.utils.**CompareOpenStackReleases** (*item*)

Bases: *charmhelpers.core.strutils.BasicStringComparator*

Provide comparisons of OpenStack releases.

Use in the form of

**if CompareOpenStackReleases(release) > 'mitaka':** # do something with mitaka

charmhelpers.contrib.openstack.utils.**check\_actually\_paused** (*services=None*,  
*ports=None*)

Check that services listed in the services object and ports are actually closed (not listened to), to verify that the unit is properly paused.

@param services: See *\_extract\_services\_list\_helper* @returns status, : string for status (None if okay)

message : string for problem for status\_set

`charmhelpers.contrib.openstack.utils.clean_storage(block_device)`

**Ensures a block device is clean. That is:**

- unmounted
- any lvm volume groups are deactivated
- any lvm physical device signatures removed
- partition table wiped

**Parameters** `block_device` – str: Full path to block device to clean.

`charmhelpers.contrib.openstack.utils.clear_unit_paused()`

Clear the unit from a paused state in the local kv() store This does NOT actually restart any services - it only clears the local state.

`charmhelpers.contrib.openstack.utils.clear_unit_upgrading()`

Clear the unit from a upgrading state in the local kv() store

`charmhelpers.contrib.openstack.utils.config_flags_parser(config_flags)`

Parses config flags string into dict.

This parsing method supports a few different formats for the config flag values to be parsed:

1. A string in the simple format of key=value pairs, with the possibility of specifying multiple key value pairs within the same string. For example, a string in the format of 'key1=value1, key2=value2' will return a dict of:

```
{'key1': 'value1', 'key2': 'value2'}
```

2. A string in the above format, but supporting a comma-delimited list of values for the same key. For example, a string in the format of 'key1=value1, key2=value3,value4,value5' will return a dict of:

```
{'key1': 'value1', 'key2': 'value2,value3,value4'}
```

3. A string containing a colon character (:) prior to an equal character (=) will be treated as yaml and parsed as such. This can be used to specify more complex key value pairs. For example, a string in the format of 'key1: subkey1=value1, subkey2=value2' will return a dict of:

```
{'key1': 'subkey1=value1, subkey2=value2'}
```

The provided `config_flags` string may be a list of comma-separated values which themselves may be comma-separated list of values.

`charmhelpers.contrib.openstack.utils.config_value_changed(option)`

Determine if config value changed since last call to this function.

`charmhelpers.contrib.openstack.utils.configure_installation_source(source_plus_key)`

Configure an installation source.

The functionality is provided by `charmhelpers.fetch.add_source()` The difference between the two functions is that `add_source()` signature requires the key to be passed directly, whereas this function passes an optional key by appending '`!<key>`' to the end of the source specification '`source`'.

Another difference from `add_source()` is that the function calls `sys.exit(1)` if the configuration fails, whereas `add_source()` raises `SourceConfigurationError()`. Another difference, is that `add_source()` silently fails (with a `juju_log` command) if there is no matching source to configure, whereas this function fails with a `sys.exit(1)`

**Parameters** `source` – String\_plus\_key – see above for details.

Note that the behaviour on error is to log the error to the juju log and then call `sys.exit(1)`.

`charmhelpers.contrib.openstack.utils.do_action_openstack_upgrade` (*package*, *upgrade\_callback*, *configs*)

Perform action-managed OpenStack upgrade.

Upgrades packages to the configured openstack-origin version and sets the corresponding action status as a result.

If the charm was installed from source we cannot upgrade it. For backwards compatibility a config flag (action-managed-upgrade) must be set for this code to run, otherwise a full service level upgrade will fire on config-changed.

@param package: package name for determining if upgrade available @param upgrade\_callback: function callback to charm's upgrade function @param configs: templating object derived from OSConfigRenderer class

@return: True if upgrade successful; False if upgrade failed or skipped

`charmhelpers.contrib.openstack.utils.enable_memcache` (*source=None*, *release=None*, *package=None*)

Determine if memcache should be enabled on the local unit

@param release: release of OpenStack currently deployed @param package: package to derive OpenStack version deployed @returns boolean Whether memcache should be enabled

`charmhelpers.contrib.openstack.utils.ensure_block_device` (*block\_device*)

Confirm *block\_device*, create as loopback if necessary.

**Parameters** *block\_device* – str: Full path of block device to ensure.

**Returns** str: Full path of ensured block device.

`charmhelpers.contrib.openstack.utils.error_out` (*msg*)

`charmhelpers.contrib.openstack.utils.get_installed_semantic_versioned_packages` ()

Get a list of installed packages which have OpenStack semantic versioning

:returns List of installed packages :rtype: [pkg1, pkg2, ...]

`charmhelpers.contrib.openstack.utils.get_matchmaker_map` (*mm\_file='/etc/oslo/matchmaker\_ring.json'*)

`charmhelpers.contrib.openstack.utils.get_os_codename_install_source` (*src*)

Derive OpenStack release codename from a given installation source.

`charmhelpers.contrib.openstack.utils.get_os_codename_package` (*package*, *fatal=True*)

Derive OpenStack release codename from an installed package.

`charmhelpers.contrib.openstack.utils.get_os_codename_version` (*vers*)

Determine OpenStack codename from version number.

```
charmhelpers.contrib.openstack.utils.get_os_version_codename(codename, version_map={
    '2011.2': 'diablo',
    '2012.1': 'essex',
    '2012.2': 'folsom',
    '2013.1': 'grizzly',
    '2013.2': 'havana',
    '2014.1': 'icehouse',
    '2014.2': 'juno',
    '2015.1': 'kilo',
    '2015.2': 'liberty',
    '2016.1': 'mitaka',
    '2016.2': 'newton',
    '2017.1': 'ocata',
    '2017.2': 'pike',
    '2018.1': 'queens',
    '2018.2': 'rocky',
    '2019.1': 'stein'})
```

Determine OpenStack version number from codename.

```
charmhelpers.contrib.openstack.utils.get_os_version_codename_swift(codename)
Determine OpenStack version number of swift from codename.
```

```
charmhelpers.contrib.openstack.utils.get_os_version_install_source(src)
charmhelpers.contrib.openstack.utils.get_os_version_package(pkg, fatal=True)
Derive OpenStack version number from an installed package.
```

```
charmhelpers.contrib.openstack.utils.get_snaps_install_info_from_origin(snaps,
    src,
    mode='classic')
```

Generate a dictionary of snap install information from origin

@param snaps: List of snaps @param src: String of openstack-origin or source of the form  
snap:track/channel

@param mode: String classic, devmode or jailmode @returns: Dictionary of snaps with channels and modes

```
charmhelpers.contrib.openstack.utils.get_source_and_pgp_key(source_and_key)
Look for a pgp key ID or ascii-armor key in the given input.
```

**Parameters** `source_and_key` – Sting, “source\_spec:keyid” where ‘!keyid’ is optional.

**:returns** (`source_spec`, `key_id` OR None) as a tuple. Returns None for `key_id` if there was no ‘!’ in the `source_and_key` string.

```
charmhelpers.contrib.openstack.utils.get_swift_codename(version)
Determine OpenStack codename that corresponds to swift version.
```

```
charmhelpers.contrib.openstack.utils.import_key(keyid)
Import a key, either ASCII armored, or a GPG key id.
```

@param keyid: the key in ASCII armor format, or a GPG key id. @raises SystemExit() via sys.exit() on failure.

```
charmhelpers.contrib.openstack.utils.incomplete_relation_data(configs,
    required_interfaces)
Check complete contexts against required_interfaces Return dictionary of incomplete relation data.
```

configs is an OSConfigRenderer object with configs registered

required\_interfaces is a dictionary of required general interfaces with dictionary values of possible specific interfaces. Example: required\_interfaces = {'database': ['shared-db', 'pgsql-db']}

The interface is said to be satisfied if anyone of the interfaces in the list has a complete context.

Return dictionary of incomplete or missing required contexts with relation status of interfaces and any missing data points. Example:

```
{'message':
  {'amqp': {'missing_data': ['rabbitmq_password'], 'related': True}, 'zeromq-
    configuration': {'related': False}},
  'identity': {'identity-service': {'related': False}},
  'database':
    {'pgsql-db': {'related': False}, 'shared-db': {'related': True}}}
```

charmhelpers.contrib.openstack.utils.install\_os\_snaps (snaps, refresh=False)  
 Install OpenStack snaps from channel and with mode

**@param snaps: Dictionary of snaps with channels and modes of the form:**

```
{'snap_name': {'channel': 'snap_channel', 'mode': 'snap_mode'}}
```

Where channel is a snapstore channel and mode is -classic, -devmode or -jailmode.

@param post\_snap\_install: Callback function to run after snaps have been installed

charmhelpers.contrib.openstack.utils.is\_unit\_paused\_set ()

Return the state of the kv().get('unit-paused'). This does NOT verify that the unit really is paused.

To help with units that don't have HookData() (testing) if it excepts, return False

charmhelpers.contrib.openstack.utils.is\_unit\_upgrading\_set ()

Return the state of the kv().get('unit-upgrading').

To help with units that don't have HookData() (testing) if it excepts, return False

charmhelpers.contrib.openstack.utils.make\_assess\_status\_func (\*args, \*\*kwargs)

Creates an assess\_status\_func() suitable for handing to pause\_unit() and resume\_unit().

This uses the \_determine\_os\_workload\_status(...) function to determine what the workload\_status should be for the unit. If the unit is not in maintenance or active states, then the message is returned to the caller. This is so an action that doesn't result in either a complete pause or complete resume can signal failure with an action\_fail()

charmhelpers.contrib.openstack.utils.manage\_payload\_services (action, services=None, charm\_func=None)

Run an action against all services.

An optional charm\_func() can be called. It should raise an Exception to indicate that the function failed. If it was succesfull it should return None or an optional message.

The signature for charm\_func is: charm\_func() -> message: str

charm\_func() is executed after any services are stopped, if supplied.

**The services object can either be:**

- None : no services were passed (an empty dict is returned)
- a list of strings

- A dictionary (optionally OrderedDict) {service\_name: {'service': ..}}
- An array of [{'service': service\_name, ...}, ...]

#### Parameters

- **action** (*str*) – Action to run: pause, resume, start or stop.
- **services** (*See above*) – See above
- **charm\_func** (*f()*) – function to run for custom charm pausing.

**Returns** Status boolean and list of messages

**Return type** (bool, [])

**Raises** RuntimeError

`charmhelpers.contrib.openstack.utils.openstack_upgrade_available(package)`  
Determines if an OpenStack upgrade is available from installation source, based on version of installed package.

**Parameters** **package** – str: Name of installed package.

**Returns** bool: : Returns True if configured installation source offers a newer version of package.

`charmhelpers.contrib.openstack.utils.ordered(orderme)`  
Converts the provided dictionary into a collections.OrderedDict.

The items in the returned OrderedDict will be inserted based on the natural sort order of the keys. Nested dictionaries will also be sorted in order to ensure fully predictable ordering.

**Parameters** **orderme** – the dict to order

**Returns** collections.OrderedDict

**Raises** ValueError: if *orderme* isn't a dict instance.

`charmhelpers.contrib.openstack.utils.os_application_version_set(package)`  
Set version of application for Juju 2.0 and later

`charmhelpers.contrib.openstack.utils.os_release(package, base='essex', re-  
set_cache=False)`

Returns OpenStack release codename from a cached global.

If *reset\_cache* then unset the cached *os\_release* version and return the freshly determined version.

If the codename can not be determined from either an installed package or the installation source, the earliest release supported by the charm should be returned.

`charmhelpers.contrib.openstack.utils.os_requires_version(ostack_release, pkg)`  
Decorator for hook to specify minimum supported release

`charmhelpers.contrib.openstack.utils.os_workload_status(configs, re-  
quired_interfaces,  
charm_func=None)`

Decorator to set workload status based on complete contexts

`charmhelpers.contrib.openstack.utils.pausable_restart_on_change(restart_map,  
stop-  
start=False,  
restart_functions=None)`

A *restart\_on\_change* decorator that checks to see if the unit is paused. If it is paused then the decorated function doesn't fire.

This is provided as a helper, as the `@restart_on_change(...)` decorator is in `core.host`, yet the openstack specific helpers are in this file (`contrib.openstack.utils`). Thus, this needs to be an optional feature for openstack charms (or charms that wish to use the openstack pause/resume type features).

It is used as follows:

```
from contrib.openstack.utils import ( pausable_restart_on_change as restart_on_change)

@restart_on_change(restart_map, stopstart=<boolean>) def some_hook(...):
    pass
```

see `core.utils.restart_on_change()` for more details.

Note `restart_map` can be a callable, in which case, `restart_map` is only evaluated at runtime. This means that it is lazy and the underlying function won't be called if the decorated function is never called. Note, retains backwards compatibility for passing a non-callable dictionary.

@param f: the function to decorate @param restart\_map: (optionally callable, which then returns the restart\_map) the restart map {conf\_file: [services]}

@param stopstart: DEFAULT false; whether to stop, start or just restart @returns decorator to use a restart\_on\_change with pausability

```
charmhelpers.contrib.openstack.utils.pause_unit (assess_status_func, services=None,
                                                ports=None, charm_func=None)
```

Pause a unit by stopping the services and setting 'unit-paused' in the local kv() store.

Also checks that the services have stopped and ports are no longer being listened to.

An optional `charm_func()` can be called that can either raise an Exception or return non None, None to indicate that the unit didn't pause cleanly.

The signature for `charm_func` is: `charm_func() -> message: string`

`charm_func()` is executed after any services are stopped, if supplied.

**The services object can either be:**

- None : no services were passed (an empty dict is returned)
- a list of strings
- A dictionary (optionally OrderedDict) {service\_name: {'service': ..}}
- An array of [{ 'service': service\_name, ... }, ... ]

@param assess\_status\_func: (f() -> message: string | None) or None @param services: OPTIONAL see above

@param ports: OPTIONAL list of port @param charm\_func: function to run for custom charm pausing. @returns None @raises Exception(message) on an error for `action_fail()`.

```
charmhelpers.contrib.openstack.utils.remote_restart (rel_name,
                                                    remote_service=None)
```

```
charmhelpers.contrib.openstack.utils.reset_os_release ()
Unset the cached os_release version
```

```
charmhelpers.contrib.openstack.utils.resume_unit (assess_status_func, services=None,
                                                  ports=None, charm_func=None)
```

Resume a unit by starting the services and clearing 'unit-paused' in the local kv() store.

Also checks that the services have started and ports are being listened to.

An optional `charm_func()` can be called that can either raise an Exception or return non None to indicate that the unit didn't resume cleanly.

The signature for `charm_func` is: `charm_func() -> message: string`



`charm_func()` is executed after any services are started, if supplied.

**The services object can either be:**

- None : no services were passed (an empty dict is returned)
- a list of strings
- A dictionary (optionally OrderedDict) {service\_name: {'service': ..}}
- An array of [{ 'service': service\_name, ... }, ... ]

@param `assess_status_func`: (f() -> message: string | None) or None @param `services`: OPTIONAL see above  
 @param `ports`: OPTIONAL list of port @param `charm_func`: function to run for custom charm resuming.  
 @returns None @raises Exception(message) on an error for `action_fail()`.

```
charmhelpers.contrib.openstack.utils.save_script_rc(script_path='scripts/scriptrc',
                                                    **env_vars)
```

Write an rc file in the charm-delivered directory containing exported environment variables provided by `env_vars`. Any charm scripts run outside the juju hook environment can source this scriptrc to obtain updated config information necessary to perform health checks or service changes.

```
charmhelpers.contrib.openstack.utils.series_upgrade_complete(resume_unit_helper=None,
                                                            configs=None)
```

Run common series upgrade complete tasks.

**Parameters**

- **resume\_unit\_helper** – function: Function to resume unit
- **configs** – OSConfigRenderer object: Configurations

**Returns None**

```
charmhelpers.contrib.openstack.utils.series_upgrade_prepare(pause_unit_helper=None,
                                                            configs=None)
```

Run common series upgrade prepare tasks.

**Parameters**

- **pause\_unit\_helper** – function: Function to pause unit
- **configs** – OSConfigRenderer object: Configurations

**Returns None**

```
charmhelpers.contrib.openstack.utils.set_os_workload_status(configs,          re-
                                                            quired_interfaces,
                                                            charm_func=None,
                                                            services=None,
                                                            ports=None)
```

Set the state of the workload status for the charm.

This calls `_determine_os_workload_status()` to get the new state, message and sets the status using `status_set()`

@param `configs`: a templating.OSConfigRenderer() object @param `required_interfaces`: {generic: [specific, specific2, ...]} @param `charm_func`: a callable function that returns state, message. The

signature is `charm_func(configs) -> (state, message)`

@param `services`: list of strings OR dictionary specifying services/ports @param `ports`: OPTIONAL list of port numbers. @returns state, message: the new workload status, user message

```
charmhelpers.contrib.openstack.utils.set_unit_paused()
```

Set the unit to a paused state in the local kv() store. This does NOT actually pause the unit

`charmhelpers.contrib.openstack.utils.set_unit_upgrading()`

Set the unit to a upgrading state in the local kv() store.

`charmhelpers.contrib.openstack.utils.snap_install_requested()`

Determine if installing from snaps

If openstack-origin is of the form snap:track/channel[/branch] and channel is in SNAPS\_CHANNELS return True.

`charmhelpers.contrib.openstack.utils.sync_db_with_multi_ipv6_addresses(database, database_user, relation_prefix=None)`

`charmhelpers.contrib.openstack.utils.token_cache_pkgs(source=None, lease=None)`

Determine additional packages needed for token caching

@param source: source string for charm @param release: release of OpenStack currently deployed @returns List of package to enable token caching

`charmhelpers.contrib.openstack.utils.update_json_file(filename, items)`

Updates the json *filename* with a given dict. :param filename: path to json file (e.g. /etc/glance/policy.json) :param items: dict of items to update

`charmhelpers.contrib.openstack.utils.workload_state_compare(current_workload_state, workload_state)`

Return highest priority of two states

### 3.2.7 charmhelpers.contrib.peerstorage package

`charmhelpers.contrib.peerstorage.leader_get(attribute=None, rid=None)`

Wrapper to ensure that settings are migrated from the peer relation.

This is to support upgrading an environment that does not support Juju leadership election to one that does.

If a setting is not extant in the leader-get but is on the relation-get peer rel, it is migrated and marked as such so that it is not re-migrated.

`charmhelpers.contrib.peerstorage.peer_echo(includes=None, force=False)`

Echo filtered attributes back onto the same relation for storage.

This is a requirement to use the peerstorage module - it needs to be called from the peer relation's changed hook.

If Juju leader support exists this will be a noop unless force is True.

`charmhelpers.contrib.peerstorage.peer_retrieve(key, relation_name='cluster')`

Retrieve a named key from peer relation *relation\_name*.

`charmhelpers.contrib.peerstorage.peer_retrieve_by_prefix(prefix, relation_name='cluster', delimiter='_', inc_list=None, exc_list=None)`

Retrieve k/v pairs given a prefix and filter using {inc,exc}\_list

`charmhelpers.contrib.peerstorage.peer_store(key, value, relation_name='cluster')`

Store the key/value pair on the named peer relation *relation\_name*.

```
charmhelpers.contrib.peerstorage.peer_store_and_set (relation_id=None,
                                                    peer_relation_name='cluster',
                                                    peer_store_fatal=False,
                                                    relation_settings=None,
                                                    delimiter='_', **kwargs)
```

Store passed-in arguments both in argument relation and in peer storage.

It functions like doing `relation_set()` and `peer_store()` at the same time, with the same data.

**@param relation\_id:** the id of the relation to store the data on. Defaults to the current relation.

**@param peer\_store\_fatal:** Set to True, the function will raise an exception should the peer storage not be available.

```
charmhelpers.contrib.peerstorage.relation_get (attribute=None, unit=None, rid=None)
Attempt to use leader-get if supported in the current version of Juju, otherwise falls back on relation-get.
```

Note that we only attempt to use leader-get if the provided rid is a peer relation id or no relation id is provided (in which case we assume we are within the peer relation context).

```
charmhelpers.contrib.peerstorage.relation_set (relation_id=None,
                                                relation_settings=None, **kwargs)
```

Attempt to use leader-set if supported in the current version of Juju, otherwise falls back on relation-set.

Note that we only attempt to use leader-set if the provided relation\_id is a peer relation id or no relation id is provided (in which case we assume we are within the peer relation context).

### 3.2.8 charmhelpers.contrib.python package

**charmhelpers.contrib.python.debug module**

**charmhelpers.contrib.python.packages module**

**charmhelpers.contrib.python.rpdb module**

Remote Python Debugger (pdb wrapper).

**charmhelpers.contrib.python.version module**

### 3.2.9 charmhelpers.contrib.saltstack package

Charm Helpers saltstack - declare the state of your machines.

This helper enables you to declare your machine state, rather than program it procedurally (and have to test each change to your procedures). Your install hook can be as simple as:

```
{ {{
from charmhelpers.contrib.saltstack import (
    install_salt_support,
    update_machine_state,
)

def install():
    install_salt_support()
    update_machine_state('machine_states/dependencies.yaml')
```

(continues on next page)

(continued from previous page)

```
update_machine_state('machine_states/installed.yaml')
}}}
```

and won't need to change (nor will its tests) when you change the machine state.

It's using a python package called salt-minion which allows various formats for specifying resources, such as:

```
{{{
/srv/{{ basedir }}:
  file.directory:
    - group: ubunet
    - user: ubunet
    - require:
      - user: ubunet
    - recurse:
      - user
      - group

ubunet:
  group.present:
    - gid: 1500
  user.present:
    - uid: 1500
    - gid: 1500
    - createhome: False
    - require:
      - group: ubunet
}}}
```

The docs for all the different state definitions are at: <http://docs.saltstack.com/ref/states/all/>

#### TODO:

- Add test helpers which will ensure that machine state definitions are functionally (but not necessarily logically) correct (ie. getting salt to parse all state defs).
- Add a link to a public bootstrap charm example / blogpost.
- Find a way to obviate the need to use the grains['charm\_dir'] syntax in templates.

`charmhelpers.contrib.saltstack.install_salt_support` (*from\_ppa=True*)  
Installs the salt-minion helper for machine state.

By default the salt-minion package is installed from the saltstack PPA. If `from_ppa` is `False` you must ensure that the salt-minion package is available in the apt cache.

`charmhelpers.contrib.saltstack.update_machine_state` (*state\_path*)  
Update the machine state using the provided state declaration.

### 3.2.10 charmhelpers.contrib.ssl package

#### charmhelpers.contrib.ssl.service module

```
class charmhelpers.contrib.ssl.service.ServiceCA (name, ca_dir, cert_type='standard')
  Bases: object

  ca_cert
  ca_conf
```

```

ca_key
create_certificate (common_name)
default_ca_expiry = '2190'
default_expiry = '730'
static get_ca (type='standard')
get_ca_bundle ()
get_certificate (common_name)
get_conf_variables ()
get_or_create_cert (common_name)
classmethod get_service_cert (type='standard')
init ()
signing_conf

```

```

charmhelpers.contrib.ssl.generate_selfsigned (keyfile, certfile, keysize='1024', con-
fig=None, subject=None, cn=None)

```

Generate selfsigned SSL keypair

You must provide one of the 3 optional arguments: config, subject or cn If more than one is provided the leftmost will be used

Arguments: keyfile – (required) full path to the keyfile to be created certfile – (required) full path to the certfile to be created keysize – (optional) SSL key length config – (optional) openssl configuration file subject – (optional) dictionary with SSL subject variables cn – (optional) certificate common name

Required keys in subject dict: cn – Common name (eq. FQDN)

Optional keys in subject dict country – Country Name (2 letter code) state – State or Province Name (full name) locality – Locality Name (eg, city) organization – Organization Name (eg, company) organizational\_unit – Organizational Unit Name (eg, section) email – Email Address

### 3.2.11 charmhelpers.contrib.storage package

#### charmhelpers.contrib.storage.linux package

#### charmhelpers.contrib.storage.linux.ceph module

```

class charmhelpers.contrib.storage.linux.ceph.CephBrokerRq (api_version=1, re-
quest_id=None)

```

Bases: object

Ceph broker request.

Multiple operations can be added to a request and sent to the Ceph broker to be executed.

Request is json-encoded for sending over the wire.

The API is versioned and defaults to version 1.

```

add_op_create_erasure_pool (name, erasure_profile=None, weight=None, group=None,
app_name=None, max_bytes=None, max_objects=None)

```

Adds an operation to create a erasure coded pool.

#### Parameters

- **name** (*str*) – Name of pool to create
- **erasure\_profile** (*str*) – Name of erasure code profile to use. If not set the ceph-mon unit handling the broker request will set its default value.
- **weight** (*float*) – The percentage of data that is expected to be contained in the pool from the total available space on the OSDs.
- **group** (*str*) – Group to add pool to
- **app\_name** (*str*) – (Optional) Tag pool with application name. Note that there is certain protocols emerging upstream with regard to meaningful application names to use. Examples are `rbd` and `rgw`.
- **max\_bytes** (*int*) – Maximum bytes quota to apply
- **max\_objects** (*int*) – Maximum objects quota to apply

**add\_op\_create\_pool** (*name*, *replica\_count*=3, *pg\_num*=None, *weight*=None, *group*=None, *namespace*=None, *app\_name*=None, *max\_bytes*=None, *max\_objects*=None)

DEPRECATED: Use `add_op_create_replicated_pool()` or `add_op_create_erasure_pool()` instead.

**add\_op\_create\_replicated\_pool** (*name*, *replica\_count*=3, *pg\_num*=None, *weight*=None, *group*=None, *namespace*=None, *app\_name*=None, *max\_bytes*=None, *max\_objects*=None)

Adds an operation to create a replicated pool.

#### Parameters

- **name** (*str*) – Name of pool to create
- **replica\_count** (*int*) – Number of copies Ceph should keep of your data.
- **pg\_num** (*int*) – Request specific number of Placement Groups to create for pool.
- **weight** (*float*) – The percentage of data that is expected to be contained in the pool from the total available space on the OSDs. Used to calculate number of Placement Groups to create for pool.
- **group** (*str*) – Group to add pool to
- **namespace** (*str*) – Group namespace
- **app\_name** (*str*) – (Optional) Tag pool with application name. Note that there is certain protocols emerging upstream with regard to meaningful application names to use. Examples are `rbd` and `rgw`.
- **max\_bytes** (*int*) – Maximum bytes quota to apply
- **max\_objects** (*int*) – Maximum objects quota to apply

**add\_op\_request\_access\_to\_group** (*name*, *namespace*=None, *permission*=None, *key\_name*=None, *object\_prefix\_permissions*=None)

Adds the requested permissions to the current service's Ceph key, allowing the key to access only the specified pools or object prefixes. `object_prefix_permissions` should be a dictionary keyed on the permission with the corresponding value being a list of prefixes to apply that permission to.

```
{ 'rwx': ['prefix1', 'prefix2'], 'class-read': ['prefix3']}
```

#### request

**set\_ops** (*ops*)

Set request ops to provided value.

Useful for injecting ops that come from a previous request to allow comparisons to ensure validity.

**class** charmhelpers.contrib.storage.linux.ceph.**CephBrokerRsp** (*encoded\_rsp*)

Bases: object

Ceph broker response.

Response is json-decoded and contents provided as methods/properties.

The API is versioned and defaults to version 1.

**exit\_code**

**exit\_msg**

**request\_id**

**class** charmhelpers.contrib.storage.linux.ceph.**CephConfContext** (*permitted\_sections=None*)

Bases: object

Ceph config (ceph.conf) context.

Supports user-provided Ceph configuration settings. Use can provide a dictionary as the value for the config-flags charm option containing Ceph configuration settings keyed by their section in ceph.conf.

**class** charmhelpers.contrib.storage.linux.ceph.**ErasurePool** (*service, name, erasure\_code\_profile='default', percent\_data=10.0, app\_name=None*)

Bases: *charmhelpers.contrib.storage.linux.ceph.Pool*

**create** ()

**class** charmhelpers.contrib.storage.linux.ceph.**Pool** (*service, name*)

Bases: object

An object oriented approach to Ceph pool creation. This base class is inherited by ReplicatedPool and ErasurePool. Do not call create() on this base class as it will not do anything. Instantiate a child class and call create().

**add\_cache\_tier** (*cache\_pool, mode*)

Adds a new cache tier to an existing pool. :param cache\_pool: six.string\_types. The cache tier pool name to add. :param mode: six.string\_types. The caching mode to use for this pool. valid range = [“readonly”, “writeback”] :return: None

**create** ()

**get\_pgs** (*pool\_size, percent\_data=10.0, device\_class=None*)

Return the number of placement groups to use when creating the pool.

Returns the number of placement groups which should be specified when creating the pool. This is based upon the calculation guidelines provided by the Ceph Placement Group Calculator (located online at <http://ceph.com/pgcalc/>).

The number of placement groups are calculated using the following:

(Pool size)

Per the upstream guidelines, the OSD # should really be considered based on the number of OSDs which are eligible to be selected by the pool. Since the pool creation doesn't specify any of CRUSH set rules, the default rule will be dependent upon the type of pool being created (replicated or erasure).

This code makes no attempt to determine the number of OSDs which can be selected for the specific rule, rather it is left to the user to tune in the form of 'expected-osd-count' config option.

### Parameters

- **pool\_size** – int. pool\_size is either the number of replicas for replicated pools or the K+M sum for erasure coded pools
- **percent\_data** – float. the percentage of data that is expected to be contained in the pool for the specific OSD set. Default value is to assume 10% of the data is for this pool, which is a relatively low % of the data but allows for the pg\_num to be increased. NOTE: the default is primarily to handle the scenario where related charms requiring pools has not been upgraded to include an update to indicate their relative usage of the pools.
- **device\_class** – str. class of storage to use for basis of pgs calculation; ceph supports nvme, ssd and hdd by default based on presence of devices of each type in the deployment.

**Returns** int. The number of pgs to use.

**remove\_cache\_tier** (*cache\_pool*)

Removes a cache tier from Ceph. Flushes all dirty objects from writeback pools and waits for that to complete. :param cache\_pool: six.string\_types. The cache tier pool name to remove. :return: None

**exception** charmhelpers.contrib.storage.linux.ceph.**PoolCreationError** (*message*)

Bases: Exception

A custom error to inform the caller that a pool creation failed. Provides an error message

**class** charmhelpers.contrib.storage.linux.ceph.**ReplicatedPool** (*service*, *name*,  
*pg\_num=None*,  
*replicas=2*, *per-*  
*cent\_data=10.0*,  
*app\_name=None*)

Bases: *charmhelpers.contrib.storage.linux.ceph.Pool*

**create** ()

charmhelpers.contrib.storage.linux.ceph.**add\_key** (*service*, *key*)

Add a key to a keyring.

Creates the keyring if it doesn't already exist.

Logs and returns if the key is already in the keyring.

charmhelpers.contrib.storage.linux.ceph.**configure** (*service*, *key*, *auth*, *use\_syslog*)

Perform basic configuration of Ceph.

charmhelpers.contrib.storage.linux.ceph.**copy\_files** (*src*, *dst*, *symlinks=False*, *ig-*  
*nore=None*)

Copy files from src to dst.

charmhelpers.contrib.storage.linux.ceph.**create\_erasure\_profile** (*service*, *pro-*  
*file\_name*, *era-*  
*sure\_plugin\_name='jerasure'*,  
*fail-*  
*ure\_domain='host'*,  
*data\_chunks=2*,  
*cod-*  
*ing\_chunks=1*,  
*locality=None*,  
*durabil-*  
*ity\_estimator=None*,  
*de-*  
*vice\_class=None*)

Create a new erasure code profile if one does not already exist for it. Updates the profile if it exists. Please see <http://docs.ceph.com/docs/master/rados/operations/erasure-code-profile/> for more details :param service: six.string\_types. The Ceph user name to run the command under :param profile\_name: six.string\_types :param



erasure\_plugin\_name: six.string\_types ;param failure\_domain: six.string\_types. One of ['chassis', 'datacenter', 'host', 'osd', 'pdu', 'pod', 'rack', 'region', 'room', 'root', 'row'])

### Parameters

- **data\_chunks** – int
- **coding\_chunks** – int
- **locality** – int
- **durability\_estimator** – int
- **device\_class** – six.string\_types

**Returns** None. Can raise CalledProcessError

charmhelpers.contrib.storage.linux.ceph.**create\_key\_file**(*service, key*)  
Create a file containing key.

charmhelpers.contrib.storage.linux.ceph.**create\_keyring**(*service, key*)  
Deprecated. Please use the more accurately named 'add\_key'

charmhelpers.contrib.storage.linux.ceph.**create\_pool**(*service, name, replicas=3, pg\_num=None*)  
Create a new RADOS pool.

charmhelpers.contrib.storage.linux.ceph.**create\_rbd\_image**(*service, pool, image, sizemb*)  
Create a new RADOS block device.

charmhelpers.contrib.storage.linux.ceph.**delete\_keyring**(*service*)  
Delete an existing Ceph keyring.

charmhelpers.contrib.storage.linux.ceph.**delete\_pool**(*service, name*)  
Delete a RADOS pool from ceph.

charmhelpers.contrib.storage.linux.ceph.**ensure\_ceph\_keyring**(*service, user=None, group=None, relation='ceph', key=None*)  
Ensures a ceph keyring is created for a named service and optionally ensures user and group ownership.

**@returns boolean: Flag to indicate whether a key was successfully written** to disk based on either relation data or a supplied key

charmhelpers.contrib.storage.linux.ceph.**ensure\_ceph\_storage**(*service, pool, rbd\_img, sizemb, mount\_point, blk\_device, fstype, system\_services=[], replicas=3*)

NOTE: This function must only be called from a single service unit for the same rbd\_img otherwise data loss will occur.

Ensures given pool and RBD image exists, is mapped to a block device, and the device is formatted and mounted at the given mount\_point.

If formatting a device for the first time, data existing at mount\_point will be migrated to the RBD device before being re-mounted.

All services listed in system\_services will be stopped prior to data migration and restarted when complete.

`charmhelpers.contrib.storage.linux.ceph.erasure_profile_exists` (*service, name*)

Check to see if an Erasure code profile already exists. :param service: six.string\_types. The Ceph user name to run the command under :param name: six.string\_types :return: int or None

`charmhelpers.contrib.storage.linux.ceph.filesystem_mounted` (*fs*)

Determine whether a filesystems is already mounted.

`charmhelpers.contrib.storage.linux.ceph.get_broker_rsp_key` ()

Return broker response key for this unit

This is the key that ceph is going to use to pass request status information back to this unit

`charmhelpers.contrib.storage.linux.ceph.get_cache_mode` (*service, pool\_name*)

Find the current caching mode of the pool\_name given. :param service: six.string\_types. The Ceph user name to run the command under :param pool\_name: six.string\_types :return: int or None

`charmhelpers.contrib.storage.linux.ceph.get_ceph_nodes` (*relation='ceph'*)

Query named relation to determine current nodes.

`charmhelpers.contrib.storage.linux.ceph.get_erasure_profile` (*service, name*)

#### Parameters

- **service** – six.string\_types. The Ceph user name to run the command under
- **name** –

#### Returns

`charmhelpers.contrib.storage.linux.ceph.get_mon_map` (*service*)

Returns the current monitor map. :param service: six.string\_types. The Ceph user name to run the command under :return: json string. :raise: ValueError if the monmap fails to parse.

Also raises CalledProcessError if our ceph command fails

`charmhelpers.contrib.storage.linux.ceph.get_osds` (*service, device\_class=None*)

Return a list of all Ceph Object Storage Daemons currently in the cluster (optionally filtered by storage device class).

**Parameters** `device_class` (*str*) – Class of storage device for OSD's

`charmhelpers.contrib.storage.linux.ceph.get_previous_request` (*rid*)

Return the last ceph broker request sent on a given relation

@param rid: Relation id to query for request

`charmhelpers.contrib.storage.linux.ceph.get_request_states` (*request, relation='ceph'*)

**Return a dict of requests per relation id with their corresponding** completion state.

This allows a charm, which has a request for ceph, to see whether there is an equivalent request already being processed and if so what state that request is in.

@param request: A CephBrokerRq object

`charmhelpers.contrib.storage.linux.ceph.hash_monitor_names` (*service*)

Uses the get\_mon\_map() function to get information about the monitor cluster. Hash the name of each monitor. Return a sorted list of monitor hashes in an ascending order. :param service: six.string\_types. The Ceph user name to run the command under :rtype : dict. json dict of monitor name, ip address and rank example: {

    'name': 'ip-172-31-13-165', 'rank': 0, 'addr': '172.31.13.165:6789/0' }

`charmhelpers.contrib.storage.linux.ceph.image_mapped` (*name*)

Determine whether a RADOS block device is mapped locally.

`charmhelpers.contrib.storage.linux.ceph.install()`

Basic Ceph client installation.

`charmhelpers.contrib.storage.linux.ceph.is_broker_action_done(action, rid=None, unit=None)`

Check whether broker action has completed yet.

@param action: name of action to be performed @returns True if action complete otherwise False

`charmhelpers.contrib.storage.linux.ceph.is_request_complete(request, relation='ceph')`

Check to see if a functionally equivalent request has already been completed

Returns True if a similar request has been completed

@param request: A CephBrokerRq object

`charmhelpers.contrib.storage.linux.ceph.is_request_complete_for_request(request, rid)`

Check if a given request has been completed on the given relation

@param request: A CephBrokerRq object @param rid: Relation ID

`charmhelpers.contrib.storage.linux.ceph.is_request_sent(request, relation='ceph')`

Check to see if a functionally equivalent request has already been sent

Returns True if a similar request has been sent

@param request: A CephBrokerRq object

`charmhelpers.contrib.storage.linux.ceph.make_filesystem(blk_device, fstype='ext4', timeout=10)`

Make a new filesystem on the specified block device.

`charmhelpers.contrib.storage.linux.ceph.map_block_storage(service, pool, image)`

Map a RADOS block device for local use.

`charmhelpers.contrib.storage.linux.ceph.mark_broker_action_done(action, rid=None, unit=None)`

Mark action as having been completed.

@param action: name of action to be performed @returns None

`charmhelpers.contrib.storage.linux.ceph.monitor_key_delete(service, key)`

Delete a key and value pair from the monitor cluster :param service: six.string\_types. The Ceph user name to run the command under Deletes a key value pair on the monitor cluster. :param key: six.string\_types. The key to delete.

`charmhelpers.contrib.storage.linux.ceph.monitor_key_exists(service, key)`

Searches for the existence of a key in the monitor cluster. :param service: six.string\_types. The Ceph user name to run the command under :param key: six.string\_types. The key to search for :return: Returns True if the key exists, False if not and raises an

exception if an unknown error occurs. :raise: CalledProcessError if an unknown error occurs

`charmhelpers.contrib.storage.linux.ceph.monitor_key_get(service, key)`

Gets the value of an existing key in the monitor cluster. :param service: six.string\_types. The Ceph user name to run the command under :param key: six.string\_types. The key to search for. :return: Returns the value of that key or None if not found.

`charmhelpers.contrib.storage.linux.ceph.monitor_key_set(service, key, value)`

Sets a key value pair on the monitor cluster. :param service: six.string\_types. The Ceph user name to run the command under :param key: six.string\_types. The key to set. :param value: The value to set. This will be converted to a string

before setting

`charmhelpers.contrib.storage.linux.ceph.place_data_on_block_device` (*blk\_device*,  
*data\_src\_dst*)

Migrate data in `data_src_dst` to `blk_device` and then remount.

`charmhelpers.contrib.storage.linux.ceph.pool_exists` (*service*, *name*)

Check to see if a RADOS pool already exists.

`charmhelpers.contrib.storage.linux.ceph.pool_set` (*service*, *pool\_name*, *key*, *value*)

Sets a value for a RADOS pool in ceph. :param `service`: six.string\_types. The Ceph user name to run the command under :param `pool_name`: six.string\_types :param `key`: six.string\_types :param `value`: :return: None. Can raise CalledProcessError

`charmhelpers.contrib.storage.linux.ceph.rbd_exists` (*service*, *pool*, *rbd\_img*)

Check to see if a RADOS block device exists.

`charmhelpers.contrib.storage.linux.ceph.remove_erasure_profile` (*service*, *profile\_name*)

Create a new erasure code profile if one does not already exist for it. Updates the profile if it exists. Please see <http://docs.ceph.com/docs/master/rados/operations/erasure-code-profile/> for more details :param `service`: six.string\_types. The Ceph user name to run the command under :param `profile_name`: six.string\_types :return: None. Can raise CalledProcessError

`charmhelpers.contrib.storage.linux.ceph.remove_pool_quota` (*service*, *pool\_name*)

Set a byte quota on a RADOS pool in ceph. :param `service`: six.string\_types. The Ceph user name to run the command under :param `pool_name`: six.string\_types :return: None. Can raise CalledProcessError

`charmhelpers.contrib.storage.linux.ceph.remove_pool_snapshot` (*service*,  
*pool\_name*, *snapshot\_name*)

Remove a snapshot from a RADOS pool in ceph. :param `service`: six.string\_types. The Ceph user name to run the command under :param `pool_name`: six.string\_types :param `snapshot_name`: six.string\_types :return: None. Can raise CalledProcessError

`charmhelpers.contrib.storage.linux.ceph.rename_pool` (*service*, *old\_name*, *new\_name*)

Rename a Ceph pool from `old_name` to `new_name` :param `service`: six.string\_types. The Ceph user name to run the command under :param `old_name`: six.string\_types :param `new_name`: six.string\_types :return: None

`charmhelpers.contrib.storage.linux.ceph.send_request_if_needed` (*request*, *relation='ceph'*)

Send broker request if an equivalent request has not already been sent

@param `request`: A CephBrokerRq object

`charmhelpers.contrib.storage.linux.ceph.set_app_name_for_pool` (*client*, *pool\_name*)

Calls `osd pool application enable` for the specified pool name

#### Parameters

- **client** (*str*) – Name of the ceph client to use
- **pool** (*str*) – Pool to set app name for
- **name** (*str*) – app name for the specified pool

**Raises** CalledProcessError if ceph call fails

`charmhelpers.contrib.storage.linux.ceph.set_pool_quota` (*service*, *pool\_name*,  
*max\_bytes=None*,  
*max\_objects=None*)

#### Parameters

- **service** (*str*) – The Ceph user name to run the command under
- **pool\_name** (*str*) – Name of pool
- **max\_bytes** (*int*) – Maximum bytes quota to apply
- **max\_objects** (*int*) – Maximum objects quota to apply

**Raises** subprocess.CalledProcessError

charmhelpers.contrib.storage.linux.ceph.**snapshot\_pool** (*service*, *pool\_name*, *snapshot\_name*)

Snapshots a RADOS pool in ceph. :param service: six.string\_types. The Ceph user name to run the command under :param pool\_name: six.string\_types :param snapshot\_name: six.string\_types :return: None. Can raise CalledProcessError

charmhelpers.contrib.storage.linux.ceph.**update\_pool** (*client*, *pool*, *settings*)

charmhelpers.contrib.storage.linux.ceph.**validator** (*value*, *valid\_type*, *valid\_range=None*)

Used to validate these: <http://docs.ceph.com/docs/master/rados/operations/pools/#set-pool-values> Example input:

```
validator(value=1, valid_type=int, valid_range=[0, 2])
```

This says I'm testing value=1. It must be an int inclusive in [0,2]

#### Parameters

- **value** – The value to validate
- **valid\_type** – The type that value should be.
- **valid\_range** – A range of values that value can assume.

#### Returns

### charmhelpers.contrib.storage.linux.loopback module

charmhelpers.contrib.storage.linux.loopback.**create\_loopback** (*file\_path*)

Create a loopback device for a given backing file.

**Returns** str: Full path to new loopback device (eg, /dev/loop0)

charmhelpers.contrib.storage.linux.loopback.**ensure\_loopback\_device** (*path*, *size*)

Ensure a loopback device exists for a given backing file path and size. If it a loopback device is not mapped to file, a new one will be created.

TODO: Confirm size of found loopback device.

**Returns** str: Full path to the ensured loopback device (eg, /dev/loop0)

charmhelpers.contrib.storage.linux.loopback.**is\_mapped\_loopback\_device** (*device*)

Checks if a given device name is an existing/mapped loopback device. :param device: str: Full path to the device (eg, /dev/loop1). :returns: str: Path to the backing file if is a loopback device empty string otherwise

charmhelpers.contrib.storage.linux.loopback.**loopback\_devices** ()

Parse through 'losetup -a' output to determine currently mapped loopback devices. Output is expected to look like:

```
/dev/loop0: [0807]:961814 (/tmp/my.img)
```

**Returns** dict: a dict mapping {loopback\_dev: backing\_file}

**charmhelpers.contrib.storage.linux.lvm module**

charmhelpers.contrib.storage.linux.lvm.**create\_logical\_volume** (*lv\_name*, *volume\_group*, *size=None*)

Create a new logical volume in an existing volume group

**Parameters**

- **lv\_name** – str: name of logical volume to be created.
- **volume\_group** – str: Name of volume group to use for the new volume.
- **size** – str: Size of logical volume to create (100% if not supplied)

**Raises** `subprocess.CalledProcessError` – in the event that the lvcreate fails.

charmhelpers.contrib.storage.linux.lvm.**create\_lvm\_physical\_volume** (*block\_device*)  
Initialize a block device as an LVM physical volume.

**Parameters** **block\_device** – str: Full path of block device to initialize.

charmhelpers.contrib.storage.linux.lvm.**create\_lvm\_volume\_group** (*volume\_group*, *block\_device*)

Create an LVM volume group backed by a given block device.

Assumes block device has already been initialized as an LVM PV.

**Parameters** **volume\_group** – str: Name of volume group to create.

**Block\_device** str: Full path of PV-initialized block device.

charmhelpers.contrib.storage.linux.lvm.**deactivate\_lvm\_volume\_group** (*block\_device*)  
Deactivate any volume group associated with an LVM physical volume.

**Parameters** **block\_device** – str: Full path to LVM physical volume

charmhelpers.contrib.storage.linux.lvm.**extend\_logical\_volume\_by\_device** (*lv\_name*, *block\_device*)

Extends the size of logical volume *lv\_name* by the amount of free space on physical volume *block\_device*.

**Parameters**

- **lv\_name** – str: name of logical volume to be extended (vg/lv format)
- **block\_device** – str: name of block\_device to be allocated to lv\_name

charmhelpers.contrib.storage.linux.lvm.**is\_lvm\_physical\_volume** (*block\_device*)  
Determine whether a block device is initialized as an LVM PV.

**Parameters** **block\_device** – str: Full path of block device to inspect.

**Returns** boolean: True if block device is a PV, False if not.

charmhelpers.contrib.storage.linux.lvm.**list\_logical\_volumes** (*select\_criteria=None*, *path\_mode=False*)

List logical volumes

**Parameters**

- **select\_criteria** – str: Limit list to those volumes matching this criteria (see ‘lvs -S help’ for more details)
- **path\_mode** – bool: return logical volume name in ‘vg/lv’ format, this format is required for some commands like lvextend

**Returns** [str]: List of logical volumes

`charmhelpers.contrib.storage.linux.lvm.list_lvm_volume_group` (*block\_device*)

List LVM volume group associated with a given block device.

Assumes block device is a valid LVM PV.

**Parameters** `block_device` – str: Full path of block device to inspect.

**Returns** str: Name of volume group associated with block device or None

`charmhelpers.contrib.storage.linux.lvm.list_thin_logical_volume_pools` (\*, *select\_criteria='lv\_attr*  
*=~*  
*^t*,  
*path\_mode=False*)

List logical volumes

**Parameters**

- **select\_criteria** – str: Limit list to those volumes matching this criteria (see ‘lvs -S help’ for more details)
- **path\_mode** – bool: return logical volume name in ‘vg/lv’ format, this format is required for some commands like lvextend

**Returns** [str]: List of logical volumes

`charmhelpers.contrib.storage.linux.lvm.list_thin_logical_volumes` (\*, *select\_criteria='lv\_attr*  
*=~* *^V*,  
*path\_mode=False*)

List logical volumes

**Parameters**

- **select\_criteria** – str: Limit list to those volumes matching this criteria (see ‘lvs -S help’ for more details)
- **path\_mode** – bool: return logical volume name in ‘vg/lv’ format, this format is required for some commands like lvextend

**Returns** [str]: List of logical volumes

`charmhelpers.contrib.storage.linux.lvm.remove_lvm_physical_volume` (*block\_device*)

Remove LVM PV signatures from a given block device.

**Parameters** `block_device` – str: Full path of block device to scrub.

### charmhelpers.contrib.storage.linux.utils module

`charmhelpers.contrib.storage.linux.utils.is_block_device` (*path*)

Confirm device at path is a valid block device node.

**Returns** boolean: True if path is a block device, False if not.

`charmhelpers.contrib.storage.linux.utils.is_device_mounted` (*device*)

Given a device path, return True if that device is mounted, and False if it isn't.

**Parameters** `device` – str: Full path of the device to check.

**Returns** boolean: True if the path represents a mounted device, False if it doesn't.

`charmhelpers.contrib.storage.linux.utils.is_luks_device` (*dev*)

Determine if dev is a LUKS-formatted block device.

**Param** dev: A full path to a block device to check for LUKS header

presence :returns: boolean: indicates whether a device is used based on LUKS header.

`charmhelpers.contrib.storage.linux.utils.is_mapped_luks_device` (*dev*)

Determine if dev is a mapped LUKS device :param: dev: A full path to a block device to be checked :returns: boolean: indicates whether a device is mapped

`charmhelpers.contrib.storage.linux.utils.mkfs_xfs` (*device, force=False*)

Format device with XFS filesystem.

By default this should fail if the device already has a filesystem on it. :param device: Full path to device to format :ptype device: tr :param force: Force operation :ptype: force: boolean

`charmhelpers.contrib.storage.linux.utils.zap_disk` (*block\_device*)

Clear a block device of partition table. Relies on sgdisk, which is installed as part of the 'gdisk' package in Ubuntu.

**Parameters** `block_device` – str: Full path of block device to clean.

## 3.2.12 charmhelpers.contrib.templating package

### charmhelpers.contrib.templating.contexts module

A helper to create a yaml cache of config with namespaced relation data.

`charmhelpers.contrib.templating.contexts.dict_keys_without_hyphens` (*a\_dict*)

Return the a new dict with underscores instead of hyphens in keys.

`charmhelpers.contrib.templating.contexts.juju_state_to_yaml` (*yaml\_path, namespace\_separator=':', allow\_hyphens\_in\_keys=True, mode=None*)

Update the juju config and state in a yaml file.

This includes any current relation-get data, and the charm directory.

This function was created for the ansible and saltstack support, as those libraries can use a yaml file to supply context to templates, but it may be useful generally to create and update an on-disk cache of all the config, including previous relation data.

By default, hyphens are allowed in keys as this is supported by yaml, but for tools like ansible, hyphens are not valid [1].

[1] [http://www.ansibleworks.com/docs/playbooks\\_variables.html#what-makes-a-valid-variable-name](http://www.ansibleworks.com/docs/playbooks_variables.html#what-makes-a-valid-variable-name)

`charmhelpers.contrib.templating.contexts.update_relations` (*context, namespace\_separator=':'*)

Update the context with the relation data.

### charmhelpers.contrib.templating.pyformat module

Templating using standard Python `str.format()` method.

`charmhelpers.contrib.templating.pyformat.render` (*template, extra={}, \*\*kwargs*)

Return the template rendered using Python's `str.format()`.



### 3.2.13 charmhelpers.contrib.unison package

`charmhelpers.contrib.unison.collect_authed_hosts` (*peer\_interface*)

Iterate through the units on peer interface to find all that have the calling host in its authorized hosts list

`charmhelpers.contrib.unison.create_private_key` (*user, priv\_key\_path, key\_type='rsa'*)

`charmhelpers.contrib.unison.create_public_key` (*user, priv\_key\_path, pub\_key\_path*)

`charmhelpers.contrib.unison.ensure_user` (*user, group=None*)

`charmhelpers.contrib.unison.get_homedir` (*user*)

`charmhelpers.contrib.unison.get_keypair` (*user*)

`charmhelpers.contrib.unison.remove_password_expiry` (*username, lastday=None, \*,  
expiredate='-1', inactive='-1',  
mindays='0', maxdays='-1',  
root=None, warndays=None*)

Change user password expiry information

#### Parameters

- **username** (*str*) – User to update
- **lastday** (*str*) – Set when password was changed in YYYY-MM-DD format
- **expiredate** (*str*) – Set when user's account will no longer be accessible in YYYY-MM-DD format. -1 will remove an account expiration date.
- **inactive** (*str*) – Set the number of days of inactivity after a password has expired before the account is locked. -1 will remove an account's inactivity.
- **mindays** (*str*) – Set the minimum number of days between password changes to MIN\_DAYS. 0 indicates the password can be changed anytime.
- **maxdays** (*str*) – Set the maximum number of days during which a password is valid. -1 as MAX\_DAYS will remove checking maxdays
- **root** (*str*) – Apply changes in the CHROOT\_DIR directory
- **warndays** (*str*) – Set the number of days of warning before a password change is required

**Raises** `subprocess.CalledProcessError` – if call to chage fails

`charmhelpers.contrib.unison.run_as_user` (*user, cmd, gid=None*)

`charmhelpers.contrib.unison.ssh_authorized_peers` (*peer\_interface, user, group=None,  
ensure\_local\_user=False*)

Main setup function, should be called from both peer -changed and -joined hooks with the same parameters.

`charmhelpers.contrib.unison.sync_path_to_host` (*path, host, user, verbose=False,  
cmd=None, gid=None, fatal=False*)

Sync path to an specific peer host

Propagates exception if operation fails and fatal=True.

`charmhelpers.contrib.unison.sync_to_peer` (*host, user, paths=None, verbose=False,  
cmd=None, gid=None, fatal=False*)

Sync paths to an specific peer host

Propagates exception if any operation fails and fatal=True.

`charmhelpers.contrib.unison.sync_to_peers` (*peer\_interface*, *user*, *paths=None*, *verbose=False*, *cmd=None*, *gid=None*, *fatal=False*)

Sync all hosts to an specific path

The type of group is integer, it allows user has permissions to operate a directory have a different group id with the user id.

Propagates exception if any operation fails and `fatal=True`.

`charmhelpers.contrib.unison.write_authorized_keys` (*user*, *keys*)

`charmhelpers.contrib.unison.write_known_hosts` (*user*, *hosts*)

### 3.3 charmhelpers.fetch package

**exception** `charmhelpers.fetch.AptLockError`

Bases: `Exception`

**class** `charmhelpers.fetch.BaseFetchHandler`

Bases: `object`

Base class for `FetchHandler` implementations in fetch plugins

**base\_url** (*url*)

Return url without querystring or fragment

**can\_handle** (*source*)

Returns True if the source can be handled. Otherwise returns a string explaining why it cannot

**install** (*source*)

Try to download and unpack the source. Return the path to the unpacked files or raise `UnhandledSource`.

**parse\_url** (*url*)

**exception** `charmhelpers.fetch.GPGKeyError`

Bases: `Exception`

Exception occurs when a GPG key cannot be fetched or used. The message indicates what the problem is.

**exception** `charmhelpers.fetch.SourceConfigError`

Bases: `Exception`

**exception** `charmhelpers.fetch.UnhandledSource`

Bases: `Exception`

`charmhelpers.fetch.configure_sources` (*update=False*, *sources\_var='install\_sources'*, *keys\_var='install\_keys'*)

Configure multiple sources from charm configuration.

The lists are encoded as yaml fragments in the configuration. The fragment needs to be included as a string. Sources and their corresponding keys are of the types supported by `add_source()`.

**Example config:**

**install\_sources:** |

- “ppa:foo”
- “http://example.com/repo precise main”

**install\_keys:** |

- null

- “a1b2c3d4”

Note that ‘null’ (a.k.a. None) should not be quoted.

`charmhelpers.fetch.install_from_config` (*config\_var\_name*)

Install a file from config.

`charmhelpers.fetch.install_remote` (*source*, *\*args*, *\*\*kwargs*)

Install a file tree from a remote source.

**The specified source should be a url of the form:** `scheme://[host]/path[#[option=value][&...]]`

Schemes supported are based on this modules submodules. Options supported are submodule-specific. Additional arguments are passed through to the submodule.

For example:

```
dest = install_remote('http://example.com/archive.tgz',
                      checksum='deadbeef',
                      hash_type='sha1')
```

This will download *archive.tgz*, validate it using SHA1 and, if the file is ok, extract it and return the directory in which it was extracted. If the checksum fails, it will raise `charmhelpers.core.host.ChecksumError`.

`charmhelpers.fetch.plugins` (*fetch\_handlers=None*)

### 3.3.1 charmhelpers.fetch.archiveurl module

#### charmhelpers.fetch.archiveurl module

**class** `charmhelpers.fetch.archiveurl.ArchiveUrlFetchHandler`

Bases: `charmhelpers.fetch.BaseFetchHandler`

Handler to download archive files from arbitrary URLs.

Can fetch from http, https, ftp, and file URLs.

Can install either tarballs (.tar, .tgz, .tbz2, etc) or zip files.

Installs the contents of the archive in `$CHARM_DIR/fetched/`.

**can\_handle** (*source*)

Returns True if the source can be handled. Otherwise returns a string explaining why it cannot

**download** (*source*, *dest*)

Download an archive file.

#### Parameters

- **source** (*str*) – URL pointing to an archive file.
- **dest** (*str*) – Local path location to download archive file to.

**download\_and\_validate** (*url*, *hashsum*, *validate='sha1'*)

**install** (*source*, *dest=None*, *checksum=None*, *hash\_type='sha1'*)

Download and install an archive file, with optional checksum validation.

The checksum can also be given on the *source* URL’s fragment. For example:

```
handler.install('http://example.com/file.tgz#sha1=deadbeef')
```

#### Parameters

- **source** (*str*) – URL pointing to an archive file.
- **dest** (*str*) – Local destination path to install to. If not given, installs to `$CHARM_DIR/archives/archive_file_name`.
- **checksum** (*str*) – If given, validate the archive file after download.
- **hash\_type** (*str*) – Algorithm used to generate *checksum*. Can be any hash algorithm supported by `hashlib`, such as `md5`, `sha1`, `sha256`, `sha512`, etc.

`charmhelpers.fetch.archiveurl.splitpasswd` (*user*)  
`urllib.splitpasswd()`, but six's support of this is missing

`charmhelpers.fetch.archiveurl.splituser` (*host*)  
`urllib.splituser()`, but six's support of this seems broken

### 3.3.2 charmhelpers.fetch.bzrurl module

`charmhelpers.fetch.bzrurl` module

### 3.3.3 charmhelpers.fetch.snap module

`charmhelpers.fetch.snap` package

Charm helpers snap for classic charms.

If writing reactive charms, use the snap layer: <https://lists.ubuntu.com/archives/snapcraft/2016-September/001114.html>

**exception** `charmhelpers.fetch.snap.CouldNotAcquireLockException`  
Bases: `Exception`

**exception** `charmhelpers.fetch.snap.InvalidSnapChannel`  
Bases: `Exception`

`charmhelpers.fetch.snap.snap_install` (*packages*, *\*flags*)  
Install a snap package.

#### Parameters

- **packages** – String or List String package name
- **flags** – List String flags to pass to install command

**Returns** Integer return code from snap

`charmhelpers.fetch.snap.snap_refresh` (*packages*, *\*flags*)  
Refresh / Update snap package.

#### Parameters

- **packages** – String or List String package name
- **flags** – List String flags to pass to refresh command

**Returns** Integer return code from snap

`charmhelpers.fetch.snap.snap_remove` (*packages*, *\*flags*)  
Remove a snap package.

#### Parameters

- **packages** – String or List String package name
- **flags** – List String flags to pass to remove command

**Returns** Integer return code from snap

`charmhelpers.fetch.snap.valid_snap_channel(channel)`

Validate snap channel exists

**Raises** *InvalidSnapChannel* – When channel does not exist

**Returns** Boolean

## Examples

```
snap_install('hello-world', '--classic', '--stable')
snap_install(['hello-world', 'htop'])
```

```
snap_refresh('hello-world', '--classic', '--stable')
snap_refresh(['hello-world', 'htop'])
```

```
snap_remove('hello-world')
snap_remove(['hello-world', 'htop'])
```

### 3.3.4 charmhelpers.fetch.python module

`charmhelpers.fetch.python` module

## 3.4 charmhelpers.payload package

### 3.4.1 charmhelpers.payload.archive module

**exception** `charmhelpers.payload.archive.ArchiveError`

Bases: `Exception`

`charmhelpers.payload.archive.archive_dest_default(archive_name)`

`charmhelpers.payload.archive.extract(archive_name, destpath=None)`

`charmhelpers.payload.archive.extract_tarfile(archive_name, destpath)`

Unpack a tar archive, optionally compressed

`charmhelpers.payload.archive.extract_zipfile(archive_name, destpath)`

Unpack a zip file

`charmhelpers.payload.archive.get_archive_handler(archive_name)`

### 3.4.2 charmhelpers.payload.execd module

`charmhelpers.payload.execd.default_execd_dir()`

`charmhelpers.payload.execd.execd_module_paths(execd_dir=None)`

Generate a list of full paths to modules within `execd_dir`.

`charmhelpers.payload.execd.execd_preinstall` (*execd\_dir=None*)

Run charm-pre-install for each module within `execd_dir`.

`charmhelpers.payload.execd.execd_run` (*command*, *execd\_dir=None*, *die\_on\_error=True*,  
*stderr=-2*)

Run `command` for each module within `execd_dir` which defines it.

`charmhelpers.payload.execd.execd_submodule_paths` (*command*, *execd\_dir=None*)

Generate a list of full paths to the specified `command` within `exec_dir`.

Tools for working with files injected into a charm just before deployment.

## 3.5 charmhelpers.cli package

### 3.5.1 charmhelpers.cli.commands module

This module loads sub-modules into the python runtime so they can be discovered via the inspect module. In order to prevent flake8 from (rightfully) telling us these are unused modules, throw a ‘ # noqa ’ at the end of each import so that the warning is suppressed.

### 3.5.2 charmhelpers.cli.host module

`charmhelpers.cli.host.mounts` ()

List mounts

**class** `charmhelpers.cli.CommandLine`

Bases: `object`

**argument\_parser** = `None`

**exit\_code** = `0`

**formatter** = `None`

**no\_output** (*decorated*)

Subcommand is not expected to return a value, so don't print a spurious None.

**run** ()

Run cli, processing arguments and executing subcommands.

**subcommand** (*command\_name=None*)

Decorate a function as a subcommand. Use its arguments as the command-line arguments

**subcommand\_builder** (*command\_name*, *description=None*)

Decorate a function that builds a subcommand. Builders should accept a single argument (the subparser instance) and return the function to be run as the command.

**subparsers** = `None`

**test\_command** (*decorated*)

Subcommand is a boolean test function, so bool return values should be converted to a 0/1 exit code.

**class** `charmhelpers.cli.OutputFormatter` (*outfile=<\_io.TextIOWrapper* *name='<stdout>'*  
*mode='w' encoding='UTF-8'>*)

Bases: `object`

**add\_arguments** (*argument\_parser*)

**csv** (*output*)

Output data as excel-compatible CSV

**format\_output** (*output*, *fmt='raw'*)

**json** (*output*)

Output data in JSON format

**py** (*output*)

Output data as a nicely-formatted python data structure

**raw** (*output*)

Output data as raw string (default)

**supported\_formats**

**tab** (*output*)

Output data in excel-compatible tab-delimited format

**yaml** (*output*)

Output data in YAML format

`charmhelpers.cli.describe_arguments` (*func*)

Analyze a function's signature and return a data structure suitable for passing in as arguments to an argparse parser's `add_argument()` method.

## 3.6 charmhelpers.coordinator package

### 3.6.1 charmhelpers.coordinator module

The coordinator module allows you to use Juju's leadership feature to coordinate operations between units of a service.

Behavior is defined in subclasses of `coordinator.BaseCoordinator`. One implementation is provided (`coordinator.Serial`), which allows an operation to be run on a single unit at a time, on a first come, first served basis. You can trivially define more complex behavior by subclassing `BaseCoordinator` or `Serial`.

**author** Stuart Bishop <[stuart.bishop@canonical.com](mailto:stuart.bishop@canonical.com)>

#### Services Framework Usage

Ensure a `peers` relation is defined in `metadata.yaml`. Instantiate a `BaseCoordinator` subclass before invoking `ServiceManager.manage()`. Ensure that `ServiceManager.manage()` is wired up to the `leader-elected`, `leader-settings-changed`, `peers relation-changed` and `peers relation-departed` hooks in addition to any other hooks you need, or your service will deadlock.

Ensure calls to `acquire()` are guarded, so that locks are only requested when they are really needed (and thus hooks only triggered when necessary). Failing to do this and calling `acquire()` unconditionally will put your unit into a hook loop. Calls to `granted()` do not need to be guarded.

For example:

```
from charmhelpers.core import hookenv, services
from charmhelpers import coordinator

def maybe_restart(servicename):
    serial = coordinator.Serial()
    if needs_restart():
        serial.acquire('restart')
    if serial.granted('restart'):
        hookenv.service_restart(servicename)
```

(continues on next page)

(continued from previous page)

```

services = [dict(service='servicename',
                 data_ready=[maybe_restart])]

if __name__ == '__main__':
    _ = coordinator.Serial() # Must instantiate before manager.manage()
    manager = services.ServiceManager(services)
    manager.manage()

```

You can implement a similar pattern using a decorator. If the lock has not been granted, an attempt to acquire() it will be made if the guard function returns True. If the lock has been granted, the decorated function is run as normal:

```

from charmhelpers.core import hookenv, services
from charmhelpers import coordinator

serial = coordinator.Serial() # Global, instantiated on module import.

def needs_restart():
    [ ... Introspect state. Return True if restart is needed ... ]

@serial.require('restart', needs_restart)
def maybe_restart(servicename):
    hookenv.service_restart(servicename)

services = [dict(service='servicename',
                 data_ready=[maybe_restart])]

if __name__ == '__main__':
    manager = services.ServiceManager(services)
    manager.manage()

```

## Traditional Usage

Ensure a peers relation is defined in metadata.yaml.

If you are using charmhelpers.core.hookenv.Hooks, ensure that a BaseCoordinator subclass is instantiated before calling Hooks.execute.

If you are not using charmhelpers.core.hookenv.Hooks, ensure that a BaseCoordinator subclass is instantiated and its handle() method called at the start of all your hooks.

For example:

```

import sys
from charmhelpers.core import hookenv
from charmhelpers import coordinator

hooks = hookenv.Hooks()

def maybe_restart():
    serial = coordinator.Serial()
    if serial.granted('restart'):
        hookenv.service_restart('myservice')

@hooks.hook
def config_changed():

```

(continues on next page)



(continued from previous page)

```

update_config()
serial = coordinator.Serial()
if needs_restart():
    serial.acquire('restart'):
    maybe_restart()

# Cluster hooks must be wired up.
@hooks.hook('cluster-relation-changed', 'cluster-relation-departed')
def cluster_relation_changed():
    maybe_restart()

# Leader hooks must be wired up.
@hooks.hook('leader-elected', 'leader-settings-changed')
def leader_settings_changed():
    maybe_restart()

[ ... repeat for *all* other hooks you are using ... ]

if __name__ == '__main__':
    _ = coordinator.Serial() # Must instantiate before execute()
    hooks.execute(sys.argv)

```

You can also use the `require` decorator. If the lock has not been granted, an attempt to `acquire()` it will be made if the guard function returns `True`. If the lock has been granted, the decorated function is run as normal:

```

from charmhelpers.core import hookenv

hooks = hookenv.Hooks()
serial = coordinator.Serial() # Must instantiate before execute()

@require('restart', needs_restart)
def maybe_restart():
    hookenv.service_restart('myservice')

@hooks.hook('install', 'config-changed', 'upgrade-charm',
           # Peers and leader hooks must be wired up.
           'cluster-relation-changed', 'cluster-relation-departed',
           'leader-elected', 'leader-settings-changed')
def default_hook():
    [...]
    maybe_restart()

if __name__ == '__main__':
    hooks.execute()

```

## Details

A simple API is provided similar to traditional locking APIs. A lock may be requested using the `acquire()` method, and the `granted()` method may be used to check if a lock previously requested by `acquire()` has been granted. It doesn't matter how many times `acquire()` is called in a hook.

Locks are released at the end of the hook they are acquired in. This may be the current hook if the unit is leader and the lock is free. It is more likely a future hook (probably `leader-settings-changed`, possibly the `peers relation-changed` or `departed` hook, potentially any hook).

Whenever a charm needs to perform a coordinated action it will `acquire()` the lock and perform the action immediately

if acquisition is successful. It will also need to perform the same action in every other hook if the lock has been granted.

## Grubby Details

Why do you need to be able to perform the same action in every hook? If the unit is the leader, then it may be able to grant its own lock and perform the action immediately in the source hook. If the unit is the leader and cannot immediately grant the lock, then its only guaranteed chance of acquiring the lock is in the peers relation-joined, relation-changed or peers relation-departed hooks when another unit has released it (the only channel to communicate to the leader is the peers relation). If the unit is not the leader, then it is unlikely the lock is granted in the source hook (a previous hook must have also made the request for this to happen). A non-leader is notified about the lock via leader settings. These changes may be visible in any hook, even before the leader-settings-changed hook has been invoked. Or the requesting unit may be promoted to leader after making a request, in which case the lock may be granted in leader-elected or in a future peers relation-changed or relation-departed hook.

This could be simpler if leader-settings-changed was invoked on the leader. We could then never grant locks except in leader-settings-changed hooks giving one place for the operation to be performed. Unfortunately this is not the case with Juju 1.23 leadership.

But of course, this doesn't really matter to most people as most people seem to prefer the Services Framework or similar reset-the-world approaches, rather than the twisty maze of attempting to deduce what should be done based on what hook happens to be running (which always seems to evolve into reset-the-world anyway when the charm grows beyond the trivial).

I chose not to implement a callback model, where a callback was passed to acquire to be executed when the lock is granted, because the callback may become invalid between making the request and the lock being granted due to an upgrade-charm being run in the interim. And it would create restrictions, such no lambdas, callback defined at the top level of a module, etc. Still, we could implement it on top of what is here, eg. by adding a defer decorator that stores a pickle of itself to disk and have BaseCoordinator unpickle and execute them when the locks are granted.

```
class charmhelpers.coordinator.BaseCoordinator (relation_key='coordinator',  
                                                peer_relation_name=None)
```

Bases: object

**acquire** (*lock*)

Acquire the named lock, non-blocking.

The lock may be granted immediately, or in a future hook.

Returns True if the lock has been granted. The lock will be automatically released at the end of the hook in which it is granted.

Do not mindlessly call this method, as it triggers a cascade of hooks. For example, if you call acquire() every time in your peers relation-changed hook you will end up with an infinite loop of hooks. It should almost always be guarded by some condition.

**grant** (*lock*, *unit*)

Maybe grant the lock to a unit.

The decision to grant the lock or not is made for \$lock by a corresponding method grant\_\$lock, which you may define in a subclass. If no such method is defined, the default\_grant method is used. See Serial.default\_grant() for details.

**granted** (*lock*)

Return True if a previously requested lock has been granted

**grants** = None

**handle** ()

**initialize** ()

**msg** (*msg*)

Emit a message. Override to customize log spam.

**released** (*unit, lock, timestamp*)

Called on the leader when it has released a lock.

By default, does nothing but log messages. Override if you need to perform additional housekeeping when a lock is released, for example recording timestamps.

**relid** = None

**relname** = None

**request\_timestamp** (*lock*)

Return the timestamp of our outstanding request for lock, or None.

Returns a `datetime.datetime()` UTC timestamp, with no `tzinfo` attribute.

**requested** (*lock*)

Return True if we are in the queue for the lock

**requests** = None

**require** (*lock, guard\_func, \*guard\_args, \*\*guard\_kw*)

Decorate a function to be run only when a lock is acquired.

The lock is requested if the guard function returns True.

The decorated function is called if the lock has been granted.

**class** `charmhelpers.coordinator.Serial` (*relation\_key='coordinator',  
peer\_relation\_name=None*)

Bases: `charmhelpers.coordinator.BaseCoordinator`

**default\_grant** (*lock, unit, granted, queue*)

Default logic to grant a lock to a unit. Unless overridden, only one unit may hold the lock and it will be granted to the earliest queued request.

To define custom logic for `$lock`, create a subclass and define a `grant_$lock` method.

*unit* is the unit name making the request.

*granted* is the set of units already granted the lock. It will never include *unit*. It may be empty.

*queue* is the list of units waiting for the lock, ordered by time of request. It will always include *unit*, but *unit* is not necessarily first.

Returns True if the lock should be granted to *unit*.

**class** `charmhelpers.coordinator.Singleton`

Bases: `type`

`charmhelpers.deprecate` (*warning, date=None, log=None*)

Add a deprecation warning the first time the function is used. The date, which is a string in semi-ISO8660 format indicate the year-month that the function is officially going to be removed.

usage:

```
@deprecate('use core/fetch/add_source() instead', '2017-04') def contributed_add_source_thing(...):
```

...

And it then prints to the log ONCE that the function is deprecated. The reason for passing the logging function (`log`) is so that `hookenv.log` can be used for a charm if needed.

### Parameters

- **warning** – String to indicate where it has moved to.
- **date** – optional string, in YYYY-MM format to indicate when the function will definitely (probably) be removed.
- **log** – The log function to call to log. If not, logs to stdout

All contributions, both code and documentation, are welcome!

### 4.1 Source

The source code is located at <https://code.launchpad.net/charm-helpers>. To submit contributions you'll need to create a Launchpad account if you do not already have one.

To get the code:

```
$ bazaar branch lp:charm-helpers
```

To build and run tests:

```
$ cd charm-helpers
$ make
```

### 4.2 Submitting a Merge Proposal

Run `make test` and ensure all tests pass. Then commit your changes and push them to a personal branch:

```
bazaar ci -m "Description of your changes"
bazaar push lp:~<launchpad-username>/charm-helpers/my-feature
```

Note that the branch name ('my-feature' in the above example) can be anything you choose - preferably something descriptive.

Once your branch is pushed, open it in a web browser, e.g.:

```
https://code.launchpad.net/~<launchpad-username>/charm-helpers/my-feature
```

Find and click on the ‘Propose for merging’ link, and on the following screen, click the ‘Propose Merge’ button.

---

**Note:** Do not set a value in the ‘Reviewer’ field - it will be set automatically.

---

## 4.3 Open Bugs

If you’re looking for something to work on, the open bug/feature list can be found at <https://bugs.launchpad.net/charm-helpers>.

## 4.4 Documentation

If you’d like to contribute to the documentation, please refer to the `HACKING.md` document in the root of the source tree for instructions on building the documentation.

Contributions to the *Examples* section of the documentation are especially welcome, and are easy to add. Simply add a new `.rst` file under `charmhelpers/docs/examples`.

## 4.5 Getting Help

If you need help you can find it in `#juju` on the Freenode IRC network. Come talk to us - we’re a friendly bunch!

### 5.1 0.19.13

Tuesday 9th April 2019

- stein: Add swift 2.21.0 (#307)
- enable disco (#306)
- Added context generator for logrotate (#303)
- Allow specifying ownership of certificate files (#302)
- Update Keystone expectations to meet security guide (#299)
- Added an “ignore” option to sysctl\_create (#300)
- Catch NoNetworkBinding for VIPs in resolve\_address (#298)
- Add LUKS helpers to charmhelpers (#296)
- Adding arch method in host (#295)

### 5.2 0.19.12

Tuesday 5th March 2019

- Use the same gpg command (#290)
- Fix openstack-upgrade-available detection to work with new versions of apt.version\_compare() (#292)

### 5.3 0.19.11

Thursday February 27 2019

- Add `getrange` command to `unitdata` CLI (#273)
- Fixing `cmp_pkgrevno` Ceph bug (#288)
- Update `swift` version for `stein` (#287)
- Add support for creating erasure coded pool and setting `max_objects` quota (#284)

### 5.4 0.19.10

Thursday February 27 2019

- Add `OpenStack` version filter to audits (#286)
- Handle new juju charm proxy settings and `https` keyserver URLs (#248)
- Allow an audit to be excluded via configuration (#282)
- Add `section-oslo-messaging-rabbit` for `Ocata+` (#283)
- Catch `NoNetworkBinding` in addition to `NotImplementedError` (#281)

### 5.5 0.19.9

Thursday February 21 2019

- Add `OpenStackSecurityGuide` auditing (#274)
- Add support for `app_name` in `add_op_create_pool` (#280)
- Update `ceph` helpers for device class support (#279)
- Remove target directory before sync (#277)
- Fix typos (#275)
- Move `contrib.python` to `fetch.python` (#272)
- Allow `None` state from `charm_func_with_configs` (#270)
- Introduce `get_distrib_codename` helper (#268)

### 5.6 0.19.8

Tuesday January 29 2019

- Add `get_installed_semantic_versioned_packages` (#269)

### 5.7 0.19.7

Saturday January 19 2019

- Fix `ceph` update keyring (#266)



## 5.8 0.19.6

Tuesday January 15 2019

- Use default sqlalchemy driver prior to stein (#264)
- nrpe: Allow services with '@' in name (#263)
- Fix a couple of docstring typos (#262)
- Use pymysql driver for mysql sqlalchemy dialect (#261)
- Separate certificates with lineseparator in bundles (#260)

## 5.9 0.19.5

Wednesday December 19 2018

- Spelling (#258)
- Dedicated VIP/CIDR fallback settings method. (#259)
- Add monitoring to vip resources in OpenStack (#257)
- Expose resource group names (#256)
- Add openstack series support for stein (#255)
- Charms can specify additional delete & group info (#253)
- Refactor vip resource creation for iface'less use (#250)
- Update copy\_nrpe\_checks() for optional c-h directory (#247)
- Extra config when generating Openstack HA settings (#249)
- Extract common code to pause/resume services (#245)
- Fix loopback devices helper for PY3 (#244)
- Add "host" option to "connect" method (#240)
- Add "proposed" to get\_os\_codename\_install\_source function (#242)
- Update amulet helper origin list for ceilometer-agent (#239)

## 5.10 0.19.4

Wednesday November 7 2018

- Consistently render haproxy.conf (#237) (#238)
- Add helpers for extracting certs from relation. (#235)
- Make the harden and pausable\_restart\_on\_change lazy (#234)
- core/host: fix changing permissions in write\_file (#233)
- Add helpers to get expected peer and related units from goal-state (#226)
- Render perms (#231)
- Add {series} support to \_add\_apt\_repository (#230)

## 5.11 0.19.3

Tuesday October 9 2018

- Adding “log” support to Neutron API context (#228)
- Enable the apache audit checks to also be PY3 compatible (#227)
- Ensure auth\_uri/auth\_url include v3 API version (#225)
- Add OpenStack context that provides versions (#224)
- Allow glance image hypervisor type to be unset (#223)
- Allow cirros image virt type to be set (#222)
- Refactor install\_ca\_cert to core.host (#220)
- Generalized glance\_create\_image (#221)
- Remove unnecessary charm relation option (#219)
- CompareHostReleases needs cosmic series support (#216)
- fetch: add helper to determine installed packages (#215)
- Quieten down unit tests (#214)
- Write all configs on series upgrade complete (#213)
- Add helpers for common series upgrade tasks (#212)
- Adding new parameters into Neutron ctxt to make NSG logging configurable (#211)
- Fix docs rendering on RTD (#210)

## 5.12 0.19.2

Monday September 10 2018

- Add helper for apt autoremove (#209)
- ensure max length of message in log func (#208)
- Add 2.19.0 to rocky swift versions (#207)
- Fix get\_ceph\_pools for mimic (#206)
- Use glance client v2 (#205)
- Support multiple WSGI vhosts in Openstack (#201)
- Series Upgrade Helpers (#200)
- Add functions for managing ssh assets in OpenStack (#197)
- Add unit\_doomed call to inform about removed units (#199)
- Rename service\_name, add helpers for model name and UUID (#196)

## 5.13 0.19.1

Wednesday July 11 2018

- Retry importing key on failure. (#194)
- Allow a src directory passed to copy\_nrpe\_checks (#193)
- Don't update updatedb.conf file if not available (#191)
- Add remaining series support for rocky (#190)
- Support multi amqp or shared-db relations in cxtxs (#188)
- LP: #1748433 Ansible version changed from 2.0 to 2.5 and there is sev... (#181)
- ovs: long interface names and existing wiring (#186)
- Add "select" function to "MySQLHelper" class (#185)

## 5.14 0.19.0

Tuesday June 5 2018

- Add set\_Open\_vSwitch\_column\_value (#182)
- update deployment to use Amulet supported storage (#183)
- Support the goal-state command (#180)

## 5.15 0.18.11

Wednesday May 16 2018

- Add support for certs relation in OpenStack charms (#173)
- Explicitly set api\_version in get\_default\_keystone\_session (#177)
- Allow forcing keystone preferred-api-version (#176)
- Retry keystone\_wait\_for\_propagation() on exception (#175)
- Revert "Adds operator.socket (#115)" (#174)
- vaultlocker: Use secret\_id's (#171)
- Reload UFW (#170)
- remove escapes from enable\_ipfix (#169)

## 5.16 0.18.9

Wednesday May 2 2018

- Adds operator.socket (#115)
- Make get\_os\_codename\_install\_source() independent of the series where it's executed (#156)
- setup.py: exclude tests and tools directories (#104)

- Support python dict in sysctl\_create (#15)
- Add notification\_format (#145)
- Enable IPFIX monitoring on OVS bridges (#168)
- Do not parse config state file if empty (#166)
- Add misc extra bits for vaultlocker work (#165)
- Update pool creation to set app-name (#163)
- Add logging of any decode Exception in config() (#161)
- Add helpers for vaultlocker (#159)
- Add support for more arguments in EnsureDirContext (#158)
- core/services : fix handling of ports (#155)
- Enable proxy header parsing (#157)
- Cache config-get data (#147)
- add\_ovsbridge\_linuxbridge fails for missing *source* in e/n/i (#153)
- Bug/1761305/ensure apache ssl (#151)

### 5.17 0.18.8

Thursday Apr 12 2018

- Allow s390x in fetch (#150)
- Read in ca certificate as binary for PY3 (#146)
- Fix keystone\_wait\_for\_propagation test helper (#144)
- Account for password field name change in PXC 5.7 (#99)
- Handle non-zero unit numbered leader (#138)
- storage: Add create\_logical\_volume helper (#141)

### 5.18 0.18.7

Monday Mar 19 2018

- Fix network get (#118)
- Fix JSON serializable error using default (#136)
- Add egress\_subnets helper to access egress-subnets on a relation (#116)
- Allow Service Manager applications to handle the ICMP protocol (#108)
- Minor fix for changelog format in docs (#134)

## 5.19 0.18.6

Thursday Mar 15 2018

- Ensure keys in cached func args are sorted (#132)
- Doc updates (#131)
- update amulet helper to fix cinder authentication with keystone v3 (#122)
- Update get\_ca to include identity-credentials (#124)
- Update IdentityService context for service\_domain\_id (#121)
- Service catalogue validators to convert to v3 (#119)
- Add code to retrieve keystone session and client (#120)
- Add 2.17.0 for queens swift versions (#117)
- Allow passing of expected number of EPs (#113)
- Add Volume API Context (#65) (#111)

## 5.20 0.18.5

Tuesday Feb 6 2018

- contrib/network: don't panic if an interface is deleted during get\_address\_in\_network (#107)
- Add string template rendering to core/templating (#102)
- Handle no network binding exception gracefully (#97)
- Support use of HAProxy context in dashboard charm (#98)
- Add from\_string template rendering capability (#87)
- add EnsureDirContext (#88)

## 5.21 0.18.4

Friday Jan 19 2018

- Fix regression in NRPE haproxy check (#95)
- Make HAProxyContext network spaces aware (#92)
- Fix file permissions on config cache and unitdata (#94)
- Fix Swift package version check (#93)
- Add helpers for hacluster interface type (#82)
- dfs: drop venv specific parts from wsgi template (#89)
- Drop OpenStack deploy-from-source helpers (#85)
- Fix for pool\_set function and validator handling of strings (#80)
- Fix presentation use of domain for identity-credentials (#79)
- Add OpenStack Context for identity-credentials interface type (#78)

- Handle profile creation in luminous (#71)
- Add support for setting object prefix permissions (#76)
- Ensure all keys checked when comparing broker obj (#75)
- Ensure json file only changed if necessary (#74)
- Update HAProxy default timeout values (#73)
- Use volumev3 for Openstack >= Pike (#65) (#66)
- Add funcs for listing & extending logical volumes (#72)
- Ceph Luminous Amulet Test Updates (#69)
- Add bionic to ubuntu host helpers (#67)
- Fix get\_swift\_codename() to work with PY3 (#62)
- Fix up ceph library exception logging for py3 (#64)
- Release: 0.18.3 (#61)

## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`





### C

charmhelpers, 95  
charmhelpers.cli, 90  
charmhelpers.cli.commands, 90  
charmhelpers.cli.host, 90  
charmhelpers.contrib, 86  
charmhelpers.contrib.ansible, 42  
charmhelpers.contrib.charmhelpers, 44  
charmhelpers.contrib.charmsupport, 47  
charmhelpers.contrib.charmsupport.nrpe, 45  
charmhelpers.contrib.charmsupport.volumes, 46  
charmhelpers.contrib.hahelpers, 49  
charmhelpers.contrib.hahelpers.apache, 47  
charmhelpers.contrib.hahelpers.cluster, 47  
charmhelpers.contrib.network, 52  
charmhelpers.contrib.network.ip, 50  
charmhelpers.contrib.network.ovs, 49  
charmhelpers.contrib.openstack, 70  
charmhelpers.contrib.openstack.alternatives, 52  
charmhelpers.contrib.openstack.context, 53  
charmhelpers.contrib.openstack.neutron, 59  
charmhelpers.contrib.openstack.templates, 52  
charmhelpers.contrib.openstack.templating, 60  
charmhelpers.contrib.openstack.utils, 62  
charmhelpers.contrib.peerstorage, 70  
charmhelpers.contrib.python, 71  
charmhelpers.contrib.python.debug, 71  
charmhelpers.contrib.python.packages, 71  
charmhelpers.contrib.python.rpdb, 71  
charmhelpers.contrib.python.version, 71  
charmhelpers.contrib.saltstack, 71  
charmhelpers.contrib.ssl, 73  
charmhelpers.contrib.ssl.service, 72  
charmhelpers.contrib.storage, 84  
charmhelpers.contrib.storage.linux, 84  
charmhelpers.contrib.storage.linux.ceph, 73  
charmhelpers.contrib.storage.linux.loopback, 81  
charmhelpers.contrib.storage.linux.lvm, 82  
charmhelpers.contrib.storage.linux.utils, 83  
charmhelpers.contrib.templating, 84  
charmhelpers.contrib.templating.contexts, 84  
charmhelpers.contrib.templating.pyformat, 84  
charmhelpers.contrib.unison, 85  
charmhelpers.coordinator, 91  
charmhelpers.core, 42  
charmhelpers.core.decorators, 13  
charmhelpers.core.fstab, 13  
charmhelpers.core.hookenv, 16  
charmhelpers.core.host, 26  
charmhelpers.core.services, 42  
charmhelpers.core.services.base, 39  
charmhelpers.core.services.helpers, 41  
charmhelpers.core.strutils, 34  
charmhelpers.core.sysctl, 34  
charmhelpers.core.templating, 34  
charmhelpers.core.unitdata, 35  
charmhelpers.fetch, 86  
charmhelpers.fetch.archiveurl, 87  
charmhelpers.fetch.python, 89  
charmhelpers.fetch.snap, 88  
charmhelpers.payload, 90  
charmhelpers.payload.archive, 89

`charmhelpers.payload.execd`, 89

## A

- acquire() (*charmhelpers.coordinator.BaseCoordinator*  
*method*), 94  
 action() (*charmhelpers.contrib.ansible.AnsibleHooks*  
*method*), 44  
 action\_fail() (in *module*  
*charmhelpers.core.hookenv*), 18  
 action\_get() (in *module*  
*charmhelpers.core.hookenv*), 18  
 action\_name() (in *module*  
*charmhelpers.core.hookenv*), 18  
 action\_set() (in *module*  
*charmhelpers.core.hookenv*), 18  
 action\_tag() (in *module*  
*charmhelpers.core.hookenv*), 18  
 action\_uuid() (in *module*  
*charmhelpers.core.hookenv*), 18  
 add() (*charmhelpers.core.fstab.Fstab* class *method*), 13  
 add\_arguments() (*charmhelpers.cli.OutputFormatter*  
*method*), 90  
 add\_bridge() (in *module*  
*charmhelpers.contrib.network.ovs*), 49  
 add\_bridge\_port() (in *module*  
*charmhelpers.contrib.network.ovs*), 49  
 add\_cache\_tier() (*charmhelpers.contrib.storage.linux.ceph.Pool*  
*method*), 75  
 add\_check() (*charmhelpers.contrib.charmsupport.nrpe.NRPE*  
*method*), 45  
 add\_entry() (*charmhelpers.core.fstab.Fstab*  
*method*), 13  
 add\_group() (in *module* *charmhelpers.core.host*), 26  
 add\_haproxy\_checks() (in *module*  
*charmhelpers.contrib.charmsupport.nrpe*),  
 45  
 add\_init\_service\_checks() (in *module*  
*charmhelpers.contrib.charmsupport.nrpe*),  
 45  
 add\_key() (in *module*  
*charmhelpers.contrib.storage.linux.ceph*),  
 76  
 add\_metric() (in *module*  
*charmhelpers.core.hookenv*), 18  
 add\_op\_create\_erasure\_pool()  
 (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRq*  
*method*), 73  
 add\_op\_create\_pool()  
 (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRq*  
*method*), 74  
 add\_op\_create\_replicated\_pool()  
 (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRq*  
*method*), 74  
 add\_op\_request\_access\_to\_group()  
 (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRq*  
*method*), 74  
 add\_ovsbridge\_linuxbridge() (in *module*  
*charmhelpers.contrib.network.ovs*), 49  
 add\_to\_updatedb\_prunepath() (in *module*  
*charmhelpers.core.host*), 26  
 add\_user\_to\_group() (in *module*  
*charmhelpers.core.host*), 27  
 adduser() (in *module* *charmhelpers.core.host*), 27  
 AMQPContext (class in  
*charmhelpers.contrib.openstack.context*),  
 53  
 AnsibleHooks (class in  
*charmhelpers.contrib.ansible*), 43  
 ApacheSSLContext (class in  
*charmhelpers.contrib.openstack.context*),  
 53  
 AppArmorContext (class in  
*charmhelpers.contrib.openstack.context*),  
 53  
 application\_name() (in *module*  
*charmhelpers.core.hookenv*), 18  
 application\_version\_set() (in *module*  
*charmhelpers.core.hookenv*), 18  
 apply\_playbook() (in *module*  
*charmhelpers.contrib.ansible*), 44  
 AptLockError, 86

archive\_dest\_default() (in module charmhelpers.payload.archive), 89

ArchiveError, 89

ArchiveUrlFetchHandler (class in charmhelpers.fetch.archiveurl), 87

argument\_parser (charmhelpers.cli.CommandLine attribute), 90

assert\_charm\_supports\_ipv6() (in module charmhelpers.contrib.network.ip), 50

atexit() (in module charmhelpers.core.hookenv), 18

atstart() (in module charmhelpers.core.hookenv), 18

## B

base\_url() (charmhelpers.fetch.BaseFetchHandler method), 86

BaseCoordinator (class in charmhelpers.coordinator), 94

BaseFetchHandler (class in charmhelpers.fetch), 86

BasicStringComparator (class in charmhelpers.core.strutils), 34

BindHostContext (class in charmhelpers.contrib.openstack.context), 54

bool\_from\_string() (in module charmhelpers.core.strutils), 34

bytes\_from\_string() (in module charmhelpers.core.strutils), 34

## C

ca\_cert (charmhelpers.contrib.ssl.service.ServiceCA attribute), 72

ca\_conf (charmhelpers.contrib.ssl.service.ServiceCA attribute), 72

ca\_key (charmhelpers.contrib.ssl.service.ServiceCA attribute), 72

cached() (in module charmhelpers.core.hookenv), 19

calico\_ctxt() (charmhelpers.contrib.openstack.context.NeutronContext method), 55

can\_handle() (charmhelpers.fetch.archiveurl.ArchiveUrlFetchHandler method), 87

can\_handle() (charmhelpers.fetch.BaseFetchHandler method), 86

canonical\_names() (charmhelpers.contrib.openstack.context.ApacheSSLContext method), 53

canonical\_url() (in module charmhelpers.contrib.hahelpers.cluster), 47

CephBrokerRq (class in charmhelpers.contrib.storage.linux.ceph), 73

CephBrokerRsp (class in charmhelpers.contrib.storage.linux.ceph), 74

CephConfContext (class in charmhelpers.contrib.storage.linux.ceph), 75

CephContext (class in charmhelpers.contrib.openstack.context), 54

chage() (in module charmhelpers.core.host), 27

changed() (charmhelpers.core.hookenv.Config method), 17

charm\_dir() (in module charmhelpers.core.hookenv), 19

charm\_name() (in module charmhelpers.core.hookenv), 19

charmhelpers (module), 95

charmhelpers.cli (module), 90

charmhelpers.cli.commands (module), 90

charmhelpers.cli.host (module), 90

charmhelpers.contrib (module), 86

charmhelpers.contrib.ansible (module), 42

charmhelpers.contrib.charmhelpers (module), 44

charmhelpers.contrib.charmsupport (module), 47

charmhelpers.contrib.charmsupport.nrpe (module), 45

charmhelpers.contrib.charmsupport.volumes (module), 46

charmhelpers.contrib.hahelpers (module), 49

charmhelpers.contrib.hahelpers.apache (module), 47

charmhelpers.contrib.hahelpers.cluster (module), 47

charmhelpers.contrib.network (module), 52

charmhelpers.contrib.network.ip (module), 50

charmhelpers.contrib.network.ovs (module), 49

charmhelpers.contrib.openstack (module), 70

charmhelpers.contrib.openstack.alternatives (module), 52

charmhelpers.contrib.openstack.context (module), 53

charmhelpers.contrib.openstack.neutron (module), 59

charmhelpers.contrib.openstack.templates (module), 52

charmhelpers.contrib.openstack.templating (module), 60

charmhelpers.contrib.openstack.utils (module), 62

charmhelpers.contrib.peerstorage (module), 70

- charmhelpers.contrib.python (module), 71
- charmhelpers.contrib.python.debug (module), 71
- charmhelpers.contrib.python.packages (module), 71
- charmhelpers.contrib.python.rpdb (module), 71
- charmhelpers.contrib.python.version (module), 71
- charmhelpers.contrib.saltstack (module), 71
- charmhelpers.contrib.ssl (module), 73
- charmhelpers.contrib.ssl.service (module), 72
- charmhelpers.contrib.storage (module), 84
- charmhelpers.contrib.storage.linux (module), 84
- charmhelpers.contrib.storage.linux.ceph (module), 73
- charmhelpers.contrib.storage.linux.loopback (module), 81
- charmhelpers.contrib.storage.linux.lvm (module), 82
- charmhelpers.contrib.storage.linux.utils (module), 83
- charmhelpers.contrib.templating (module), 84
- charmhelpers.contrib.templating.contexts (module), 84
- charmhelpers.contrib.templating.pyformat (module), 84
- charmhelpers.contrib.unison (module), 85
- charmhelpers.coordinator (module), 91
- charmhelpers.core (module), 42
- charmhelpers.core.decorators (module), 13
- charmhelpers.core.fstab (module), 13
- charmhelpers.core.hookenv (module), 16
- charmhelpers.core.host (module), 26
- charmhelpers.core.services (module), 42
- charmhelpers.core.services.base (module), 39
- charmhelpers.core.services.helpers (module), 41
- charmhelpers.core.strutils (module), 34
- charmhelpers.core.sysctl (module), 34
- charmhelpers.core.templating (module), 34
- charmhelpers.core.unitdata (module), 35
- charmhelpers.fetch (module), 86
- charmhelpers.fetch.archiveurl (module), 87
- charmhelpers.fetch.python (module), 89
- charmhelpers.fetch.snap (module), 88
- charmhelpers.payload (module), 90
- charmhelpers.payload.archive (module), 89
- charmhelpers.payload.execd (module), 89
- chdir () (in module charmhelpers.core.host), 27
- Check (class in charmhelpers.contrib.charmsupport.nrpe), 45
- check\_actually\_paused () (in module charmhelpers.contrib.openstack.utils), 62
- check\_for\_eni\_source () (in module charmhelpers.contrib.network.ovs), 49
- check\_hash () (in module charmhelpers.core.host), 28
- CheckException, 45
- ChecksumError, 26
- chownr () (in module charmhelpers.core.host), 28
- clean\_storage () (in module charmhelpers.contrib.openstack.utils), 62
- clear\_unit\_paused () (in module charmhelpers.contrib.openstack.utils), 63
- clear\_unit\_upgrading () (in module charmhelpers.contrib.openstack.utils), 63
- close () (charmhelpers.core.unitdata.Storage method), 38
- close\_port () (in module charmhelpers.core.hookenv), 19
- close\_ports () (in module charmhelpers.core.hookenv), 19
- collect\_authed\_hosts () (in module charmhelpers.contrib.unison), 85
- CommandLine (class in charmhelpers.cli), 90
- CompareOpenStackReleases (class in charmhelpers.contrib.openstack.utils), 62
- complete (charmhelpers.contrib.openstack.context.OSContextGenerator attribute), 56
- complete\_contexts () (charmhelpers.contrib.openstack.templating.OSConfigRenderer method), 61
- complete\_contexts () (charmhelpers.contrib.openstack.templating.OSConfigTemplate method), 62
- Config (class in charmhelpers.core.hookenv), 16
- config () (in module charmhelpers.core.hookenv), 19
- CONFIG\_FILE\_NAME (charmhelpers.core.hookenv.Config attribute), 17
- config\_flags\_parser () (in module charmhelpers.contrib.openstack.utils), 63
- config\_value\_changed () (in module charmhelpers.contrib.openstack.utils), 63
- configure () (in module charmhelpers.contrib.storage.linux.ceph), 76
- configure\_ca () (charmhelpers.contrib.openstack.context.ApacheSSL method), 53
- configure\_cert () (charmhelpers.contrib.openstack.context.ApacheSSL method), 53
- configure\_installation\_source () (in module charmhelpers.contrib.openstack.utils), 63

`configure_sources()` (in module `charmhelpers.contrib.unison`), 85  
`charmhelpers.fetch`), 86  
`configure_volume()` (in module `charmhelpers.contrib.unison`), 85  
`charmhelpers.contrib.charmsupport.volumes`), 47  
`context()` (`charmhelpers.contrib.openstack.templating.OSConfigTemplate` method), 62  
`context_complete()` (`charmhelpers.contrib.openstack.context.OSContextGenerator` method), 56  
`context_complete()` (in module `charmhelpers.contrib.openstack.context`), 58  
`copy_files()` (in module `charmhelpers.contrib.storage.linux.ceph`), 76  
`copy_nrpe_checks()` (in module `charmhelpers.contrib.charmsupport.nrpe`), 45  
`CouldNotAcquireLockException`, 88  
`create()` (`charmhelpers.contrib.storage.linux.ceph.ErasurePool` method), 75  
`create()` (`charmhelpers.contrib.storage.linux.ceph.Pool` method), 75  
`create()` (`charmhelpers.contrib.storage.linux.ceph.ReplicatedPool` method), 76  
`create()` (in module `charmhelpers.core.sysctl`), 34  
`create_certificate()` (`charmhelpers.contrib.ssl.service.ServiceCA` method), 73  
`create_erasure_profile()` (in module `charmhelpers.contrib.storage.linux.ceph`), 76  
`create_key_file()` (in module `charmhelpers.contrib.storage.linux.ceph`), 77  
`create_keyring()` (in module `charmhelpers.contrib.storage.linux.ceph`), 77  
`create_logical_volume()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82  
`create_loopback()` (in module `charmhelpers.contrib.storage.linux.loopback`), 81  
`create_lvm_physical_volume()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82  
`create_lvm_volume_group()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82  
`create_pool()` (in module `charmhelpers.contrib.storage.linux.ceph`), 77  
`create_private_key()` (in module `charmhelpers.contrib.unison`), 85  
`create_public_key()` (in module `charmhelpers.contrib.unison`), 85  
`create_rbd_image()` (in module `charmhelpers.contrib.storage.linux.ceph`), 47  
`CRMDNotFound`, 47  
`CRMResourceNotFound`, 47  
`ctxt` (`charmhelpers.contrib.openstack.context.AppArmorContext` attribute), 53  
`ctxt` (`charmhelpers.contrib.openstack.context.VolumeAPIContext` attribute), 58  
`current` (`charmhelpers.core.unitdata.Delta` attribute), 37

## D

`DataPortContext` (class in `charmhelpers.contrib.openstack.context`), 54  
`delete_pool()` (in module `charmhelpers.contrib.storage.linux.ceph`), 77  
`deactivate_lvm_volume_group()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82  
`debug()` (`charmhelpers.core.unitdata.Storage` method), 38  
`default_ca_expiry` (`charmhelpers.contrib.ssl.service.ServiceCA` attribute), 73  
`default_execd_dir()` (in module `charmhelpers.payload.execd`), 89  
`default_expiry` (`charmhelpers.contrib.ssl.service.ServiceCA` attribute), 73  
`default_grant()` (`charmhelpers.coordinator.Serial` method), 95  
`DEFAULT_PATH` (`charmhelpers.core.fstab.Fstab` attribute), 13  
`del_bridge()` (in module `charmhelpers.contrib.network.ovs`), 49  
`del_bridge_port()` (in module `charmhelpers.contrib.network.ovs`), 49  
`delete_keyring()` (in module `charmhelpers.contrib.storage.linux.ceph`), 77  
`delete_pool()` (in module `charmhelpers.contrib.storage.linux.ceph`), 77  
`Delta` (class in `charmhelpers.core.unitdata`), 37  
`delta()` (`charmhelpers.core.unitdata.Storage` method), 38  
`DeltaSet` (class in `charmhelpers.core.unitdata`), 37  
`deprecate()` (in module `charmhelpers`), 95



- describe\_arguments() (in module *charmhelpers.cli*), 91  
 determine\_apache\_port() (in module *charmhelpers.contrib.hahelpers.cluster*), 47  
 determine\_api\_port() (in module *charmhelpers.contrib.hahelpers.cluster*), 48  
 determine\_dkms\_package() (in module *charmhelpers.contrib.openstack.neutron*), 59  
 dict\_keys\_without\_hyphens() (in module *charmhelpers.contrib.templating.contexts*), 84  
 disable\_ipfix() (in module *charmhelpers.contrib.network.ovs*), 49  
 distributed\_wait() (in module *charmhelpers.contrib.hahelpers.cluster*), 48  
 do\_action\_openstack\_upgrade() (in module *charmhelpers.contrib.openstack.utils*), 63  
 download() (*charmhelpers.fetch.archiveurl.ArchiveUrlFetchHandler* method), 87  
 download\_and\_validate() (*charmhelpers.fetch.archiveurl.ArchiveUrlFetchHandler* method), 87
- ## E
- egress\_subnets() (in module *charmhelpers.core.hookenv*), 19  
 eligible\_leader() (in module *charmhelpers.contrib.hahelpers.cluster*), 48  
 enable\_ipfix() (in module *charmhelpers.contrib.network.ovs*), 49  
 enable\_memcache() (in module *charmhelpers.contrib.openstack.utils*), 64  
 enable\_modules() (*charmhelpers.contrib.openstack.context.ApacheSSLContext* method), 53  
 ensure\_block\_device() (in module *charmhelpers.contrib.openstack.utils*), 64  
 ensure\_ceph\_keyring() (in module *charmhelpers.contrib.storage.linux.ceph*), 77  
 ensure\_ceph\_storage() (in module *charmhelpers.contrib.storage.linux.ceph*), 77  
 ensure\_loopback\_device() (in module *charmhelpers.contrib.storage.linux.loopback*), 81  
 ensure\_packages() (in module *charmhelpers.contrib.openstack.context*), 58  
 ensure\_user() (in module *charmhelpers.contrib.unison*), 85  
 EnsureDirContext (class in *charmhelpers.contrib.openstack.context*), 54  
 entries (*charmhelpers.core.fstab.Fstab* attribute), 13  
 env\_proxy\_settings() (in module *charmhelpers.core.hookenv*), 20  
 erasure\_profile\_exists() (in module *charmhelpers.contrib.storage.linux.ceph*), 78  
 ErasurePool (class in *charmhelpers.contrib.storage.linux.ceph*), 75  
 error\_out() (in module *charmhelpers.contrib.openstack.utils*), 64  
 execd\_module\_paths() (in module *charmhelpers.payload.execd*), 89  
 execd\_preinstall() (in module *charmhelpers.payload.execd*), 89  
 execd\_run() (in module *charmhelpers.payload.execd*), 90  
 execd\_module\_paths() (in module *charmhelpers.payload.execd*), 90  
 execute() (*charmhelpers.contrib.ansible.AnsibleHooks* method), 44  
 execute() (*charmhelpers.core.hookenv.Hooks* method), 17  
 execution\_environment() (in module *charmhelpers.core.hookenv*), 20  
 exit\_code (*charmhelpers.cli.CommandLine* attribute), 90  
 exit\_code (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRsp* attribute), 75  
 exit\_msg (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRsp* attribute), 75  
 expected\_peer\_units() (in module *charmhelpers.core.hookenv*), 20  
 expected\_ssl\_context() (in module *charmhelpers.core.hookenv*), 20  
 extend\_logical\_volume\_by\_device() (in module *charmhelpers.contrib.storage.linux.lvm*), 82  
 external\_ports (*charmhelpers.contrib.openstack.context.ApacheSSLContext* attribute), 53  
 ExternalPortContext (class in *charmhelpers.contrib.openstack.context*), 54  
 extract() (in module *charmhelpers.payload.archive*), 89  
 extract\_tarfile() (in module *charmhelpers.payload.archive*), 89  
 extract\_zipfile() (in module *charmhelpers.payload.archive*), 89

## F

- `file_hash()` (in module `charmhelpers.core.host`), 28
- `filesystem_mounted()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78
- `fire_event()` (`charmhelpers.core.services.base.ServiceManager` method), 39
- `flush()` (`charmhelpers.core.unitdata.Storage` method), 38
- `flush()` (in module `charmhelpers.core.hookenv`), 20
- `format_ipv6_addr()` (in module `charmhelpers.contrib.network.ip`), 50
- `format_output()` (`charmhelpers.cli.OutputFormatter` method), 91
- `formatter` (`charmhelpers.cli.CommandLine` attribute), 90
- `Fstab` (class in `charmhelpers.core.fstab`), 13
- `Fstab.Entry` (class in `charmhelpers.core.fstab`), 13
- `fstab_add()` (in module `charmhelpers.core.host`), 28
- `fstab_mount()` (in module `charmhelpers.core.host`), 28
- `fstab_remove()` (in module `charmhelpers.core.host`), 28
- `full_restart()` (in module `charmhelpers.contrib.network.ovs`), 49
- G**
- `generate_selfsigned()` (in module `charmhelpers.contrib.ssl`), 73
- `get()` (`charmhelpers.core.unitdata.Storage` method), 38
- `get_address_in_network()` (in module `charmhelpers.contrib.network.ip`), 50
- `get_archive_handler()` (in module `charmhelpers.payload.archive`), 89
- `get_bond_master()` (in module `charmhelpers.core.host`), 28
- `get_bridge_nics()` (in module `charmhelpers.contrib.network.ip`), 50
- `get_bridges()` (in module `charmhelpers.contrib.network.ip`), 50
- `get_broker_rsp_key()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78
- `get_ca()` (`charmhelpers.contrib.ssl.service.ServiceCA` static method), 73
- `get_ca_bundle()` (`charmhelpers.contrib.ssl.service.ServiceCA` method), 73
- `get_ca_cert()` (in module `charmhelpers.contrib.hahelpers.apache`), 47
- `get_cache_mode()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78
- `get_ceph_nodes()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78
- `get_cert()` (in module `charmhelpers.contrib.hahelpers.apache`), 47
- `get_certificate()` (`charmhelpers.contrib.ssl.service.ServiceCA` method), 73
- `get_certificate()` (in module `charmhelpers.contrib.network.ovs`), 49
- `get_conf_variables()` (`charmhelpers.contrib.ssl.service.ServiceCA` method), 73
- `get_config()` (in module `charmhelpers.contrib.charmsupport.volumes`), 47
- `get_data()` (`charmhelpers.core.services.helpers.RelationContext` method), 41
- `get_entry_by_attr()` (`charmhelpers.core.fstab.Fstab` method), 13
- `get_erasure_profile()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78
- `get_hacluster_config()` (in module `charmhelpers.contrib.hahelpers.cluster`), 48
- `get_homedir()` (in module `charmhelpers.contrib.unison`), 85
- `get_host_ip()` (in module `charmhelpers.contrib.network.ip`), 50
- `get_hostname()` (in module `charmhelpers.contrib.network.ip`), 50
- `get_iface_addr()` (in module `charmhelpers.contrib.network.ip`), 50
- `get_iface_for_address()` (in module `charmhelpers.contrib.network.ip`), 51
- `get_iface_from_addr()` (in module `charmhelpers.contrib.network.ip`), 51
- `get_incomplete_context_data()` (`charmhelpers.contrib.openstack.templating.OSConfigRenderer` method), 61
- `get_installed_semantic_versioned_packages()` (in module `charmhelpers.contrib.openstack.utils`), 64
- `get_ipv4_addr()` (in module `charmhelpers.contrib.network.ip`), 51
- `get_ipv4_addr()` (in module `charmhelpers.contrib.openstack.context`), 59
- `get_ipv6_addr()` (in module `charmhelpers.contrib.network.ip`), 51
- `get_keypair()` (in module



*charmhelpers.contrib.unison*), 85  
get\_loader() (in module *charmhelpers.contrib.openstack.templating*), 62  
get\_matchmaker\_map() (in module *charmhelpers.contrib.openstack.utils*), 64  
get\_mon\_map() (in module *charmhelpers.contrib.storage.linux.ceph*), 78  
get\_nagios\_hostcontext() (in module *charmhelpers.contrib.charmsupport.nrpe*), 45  
get\_nagios\_hostname() (in module *charmhelpers.contrib.charmsupport.nrpe*), 46  
get\_nagios\_unit\_name() (in module *charmhelpers.contrib.charmsupport.nrpe*), 46  
get\_netmask\_for\_address() (in module *charmhelpers.contrib.network.ip*), 51  
get\_netmask\_for\_address() (in module *charmhelpers.contrib.openstack.context*), 59  
get\_network\_addresses() (*charmhelpers.contrib.openstack.context.ApacheSSLContext* method), 53  
get\_neutron\_options() (*charmhelpers.contrib.openstack.context.NeutronAPIContext* method), 55  
get\_nic\_hwaddr() (in module *charmhelpers.core.host*), 28  
get\_nic\_mtu() (in module *charmhelpers.core.host*), 28  
get\_or\_create\_cert() (*charmhelpers.contrib.ssl.service.ServiceCA* method), 73  
get\_os\_codename\_install\_source() (in module *charmhelpers.contrib.openstack.utils*), 64  
get\_os\_codename\_package() (in module *charmhelpers.contrib.openstack.utils*), 64  
get\_os\_codename\_version() (in module *charmhelpers.contrib.openstack.utils*), 64  
get\_os\_version\_codename() (in module *charmhelpers.contrib.openstack.utils*), 64  
get\_os\_version\_codename\_swift() (in module *charmhelpers.contrib.openstack.utils*), 65  
get\_os\_version\_install\_source() (in module *charmhelpers.contrib.openstack.utils*), 65  
get\_os\_version\_package() (in module *charmhelpers.contrib.openstack.utils*), 65  
get\_osds() (in module *charmhelpers.contrib.storage.linux.ceph*), 78  
get\_pgsql() (*charmhelpers.contrib.storage.linux.ceph.Pool* method), 75  
get\_previous\_request() (in module *charmhelpers.contrib.storage.linux.ceph*), 78  
get\_related() (*charmhelpers.contrib.openstack.context.OSContextGenerator* method), 56  
get\_relation\_ip() (in module *charmhelpers.contrib.network.ip*), 51  
get\_request\_states() (in module *charmhelpers.contrib.storage.linux.ceph*), 78  
get\_service() (*charmhelpers.core.services.base.ServiceManager* method), 39  
get\_service\_cert() (*charmhelpers.contrib.ssl.service.ServiceCA* class method), 73  
get\_snaps\_install\_info\_from\_origin() (in module *charmhelpers.contrib.openstack.utils*), 65  
get\_source\_and\_pgp\_key() (in module *charmhelpers.contrib.openstack.utils*), 65  
get\_swift\_codename() (in module *charmhelpers.contrib.openstack.utils*), 65  
get\_total\_ram() (in module *charmhelpers.core.host*), 28  
gethistory() (*charmhelpers.core.unitdata.StorageAPIContext* method), 38  
getunit() (*charmhelpers.core.unitdata.StorageAPIContext* method), 38  
gid\_exists() (in module *charmhelpers.core.host*), 28  
goal\_state() (in module *charmhelpers.core.hookenv*), 20  
GPGKeyError, 86  
grant() (*charmhelpers.coordinator.BaseCoordinator* method), 94  
granted() (*charmhelpers.coordinator.BaseCoordinator* method), 94  
grants (*charmhelpers.coordinator.BaseCoordinator* attribute), 94  
group (*charmhelpers.contrib.openstack.context.ApacheSSLContext* attribute), 53  
group\_exists() (in module *charmhelpers.core.host*), 29

## H

HAIncompleteConfig, 47  
HAINcorrectConfig, 47  
handle() (*charmhelpers.coordinator.BaseCoordinator* method), 94  
HAProxyContext (class in *charmhelpers.contrib.openstack.context*), 54

`has_juju_version()` (in module `charmhelpers.core.hookenv`), 21  
`hash_monitor_names()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78  
`headers_package()` (in module `charmhelpers.contrib.openstack.neutron`), 59  
`homedir()` (`charmhelpers.contrib.charmsupport.nrpe.NRPE` attribute), 45  
`hook()` (`charmhelpers.core.hookenv.Hooks` method), 17  
`hook_name()` (in module `charmhelpers.core.hookenv`), 21  
`hook_scope()` (`charmhelpers.core.unitdata.Storage` method), 38  
`HookData` (class in `charmhelpers.core.unitdata`), 37  
`Hooks` (class in `charmhelpers.core.hookenv`), 17  
`https()` (in module `charmhelpers.contrib.hahelpers.cluster`), 48

I

`IdentityCredentialsContext` (class in `charmhelpers.contrib.openstack.context`), 54  
`IdentityServiceContext` (class in `charmhelpers.contrib.openstack.context`), 55  
`image_mapped()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78  
`ImageServiceContext` (class in `charmhelpers.contrib.openstack.context`), 55  
`import_key()` (in module `charmhelpers.contrib.openstack.utils`), 65  
`in_relation_hook()` (in module `charmhelpers.core.hookenv`), 21  
`incomplete_relation_data()` (in module `charmhelpers.contrib.openstack.utils`), 65  
`ingress_address()` (in module `charmhelpers.core.hookenv`), 21  
`init()` (`charmhelpers.contrib.ssl.service.ServiceCA` method), 73  
`init_is_systemd()` (in module `charmhelpers.core.host`), 29  
`initialize()` (`charmhelpers.coordinator.BaseCoordinator` method), 94  
`install()` (`charmhelpers.fetch.archiveurl.ArchiveUrlFetchHandler` method), 87  
`install()` (`charmhelpers.fetch.BaseFetchHandler` method), 86  
`install()` (in module `charmhelpers.contrib.storage.linux.ceph`), 78  
`install_aa_utils()` (`charmhelpers.contrib.openstack.context.AppArmorContext` method), 53  
`install_alternative()` (in module `charmhelpers.contrib.openstack.alternatives`), 52  
`install_ansible_support()` (in module `charmhelpers.contrib.ansible`), 44  
`install_ca_cert()` (in module `charmhelpers.contrib.hahelpers.apache`), 47  
`install_ca_cert()` (in module `charmhelpers.core.host`), 29  
`install_from_config()` (in module `charmhelpers.fetch`), 87  
`install_os_snaps()` (in module `charmhelpers.contrib.openstack.utils`), 66  
`install_remote()` (in module `charmhelpers.fetch`), 87  
`install_salt_support()` (in module `charmhelpers.contrib.saltstack`), 72  
`interface` (`charmhelpers.core.services.helpers.RelationContext` attribute), 41  
`interface_to_relations()` (in module `charmhelpers.core.hookenv`), 21  
`interfaces` (`charmhelpers.contrib.openstack.context.ApacheSSLContext` attribute), 53  
`interfaces` (`charmhelpers.contrib.openstack.context.CephContext` attribute), 54  
`interfaces` (`charmhelpers.contrib.openstack.context.HAProxyContext` attribute), 54  
`interfaces` (`charmhelpers.contrib.openstack.context.ImageServiceContext` attribute), 55  
`interfaces` (`charmhelpers.contrib.openstack.context.NeutronAPIContext` attribute), 55  
`interfaces` (`charmhelpers.contrib.openstack.context.NeutronContext` attribute), 55  
`interfaces` (`charmhelpers.contrib.openstack.context.OSContextGenerator` attribute), 56  
`interfaces` (`charmhelpers.contrib.openstack.context.PostgresqlDBContext` attribute), 57  
`interfaces` (`charmhelpers.contrib.openstack.context.SharedDBContext` attribute), 57  
`interfaces` (`charmhelpers.contrib.openstack.context.ZeroMQContext` attribute), 58  
`InternalEndpointContext` (class in `charmhelpers.contrib.openstack.context`), 55  
`InvalidSnapChannel`, 88  
`is_address_in_network()` (in module `charmhelpers.contrib.network.ip`), 51  
`is_block_device()` (in module `charmhelpers.contrib.storage.linux.utils`), 78

- 83
- `is_bridge_member()` (in module `charmhelpers.contrib.network.ip`), 52
- `is_broker_action_done()` (in module `charmhelpers.contrib.storage.linux.ceph`), 79
- `is_clustered()` (in module `charmhelpers.contrib.hahelpers.cluster`), 48
- `is_container()` (in module `charmhelpers.core.host`), 29
- `is_crm_dc()` (in module `charmhelpers.contrib.hahelpers.cluster`), 48
- `is_device_mounted()` (in module `charmhelpers.contrib.storage.linux.utils`), 83
- `is_elected_leader()` (in module `charmhelpers.contrib.hahelpers.cluster`), 48
- `is_ip()` (in module `charmhelpers.contrib.network.ip`), 52
- `is_ipv6()` (in module `charmhelpers.contrib.network.ip`), 52
- `is_ipv6_disabled()` (in module `charmhelpers.contrib.network.ip`), 52
- `is_leader()` (in module `charmhelpers.contrib.hahelpers.cluster`), 48
- `is_leader()` (in module `charmhelpers.core.hookenv`), 21
- `is_linuxbridge_interface()` (in module `charmhelpers.contrib.network.ovs`), 49
- `is_luks_device()` (in module `charmhelpers.contrib.storage.linux.utils`), 83
- `is_lvm_physical_volume()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82
- `is_mapped_loopback_device()` (in module `charmhelpers.contrib.storage.linux.loopback`), 81
- `is_mapped_luks_device()` (in module `charmhelpers.contrib.storage.linux.utils`), 84
- `is_phy_iface()` (in module `charmhelpers.core.host`), 29
- `is_ready()` (`charmhelpers.core.services.base.ServiceManager` method), 39
- `is_ready()` (`charmhelpers.core.services.helpers.RelationContext` method), 41
- `is_relation_made()` (in module `charmhelpers.core.hookenv`), 21
- `is_request_complete()` (in module `charmhelpers.contrib.storage.linux.ceph`), 79
- `is_request_complete_for_rid()` (in module `charmhelpers.contrib.storage.linux.ceph`), 79
- `is_request_sent()` (in module `charmhelpers.contrib.storage.linux.ceph`), 79
- `is_string_template` (`charmhelpers.contrib.openstack.templating.OSConfigTemplate` attribute), 62
- `is_unit_paused_set()` (in module `charmhelpers.contrib.openstack.utils`), 66
- `is_unit_upgrading_set()` (in module `charmhelpers.contrib.openstack.utils`), 66
- `iter_units_for_relation_name()` (in module `charmhelpers.core.hookenv`), 21
- ## J
- `json()` (`charmhelpers.cli.OutputFormatter` method), 91
- `json()` (`charmhelpers.core.hookenv.Serializable` method), 18
- `juju_state_to_yaml()` (in module `charmhelpers.contrib.templating.contexts`), 84
- `juju_version()` (in module `charmhelpers.core.hookenv`), 22
- ## K
- `kernel_version()` (in module `charmhelpers.contrib.openstack.neutron`), 59
- `kv()` (in module `charmhelpers.core.unitdata`), 38
- ## L
- `lchownr()` (in module `charmhelpers.core.host`), 29
- `leader_get()` (in module `charmhelpers.contrib.peerstorage`), 70
- `leader_get()` (in module `charmhelpers.core.hookenv`), 22
- `leader_set()` (in module `charmhelpers.core.hookenv`), 22
- `LibvirtConfigFlagsContext` (class in `charmhelpers.contrib.openstack.context`), 55
- `list_logical_volumes()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82
- `list_lvm_volume_group()` (in module `charmhelpers.contrib.storage.linux.lvm`), 82
- `list_nics()` (in module `charmhelpers.core.host`), 29
- `list_thin_logical_volume_pools()` (in module `charmhelpers.contrib.storage.linux.lvm`), 83

list\_thin\_logical\_volumes() (in module charmhelpers.contrib.storage.linux.lvm), 83

load\_previous() (charmhelpers.core.hookenv.Config method), 17

local\_unit() (in module charmhelpers.core.hookenv), 22

log() (in module charmhelpers.core.hookenv), 22

LogLevelContext (class in charmhelpers.contrib.openstack.context), 55

LogrotateContext (class in charmhelpers.contrib.openstack.context), 55

loopback\_devices() (in module charmhelpers.contrib.storage.linux.loopback), 81

## M

make\_assess\_status\_func() (in module charmhelpers.contrib.openstack.utils), 66

make\_filesystem() (in module charmhelpers.contrib.storage.linux.ceph), 79

manage() (charmhelpers.core.services.base.ServiceManager method), 39

manage\_payload\_services() (in module charmhelpers.contrib.openstack.utils), 66

managed\_mounts() (in module charmhelpers.contrib.charmsupport.volumes), 47

ManagerCallback (class in charmhelpers.core.services.base), 40

manually\_disable\_aa\_profile() (charmhelpers.contrib.openstack.context.AppArmorContext method), 53

map\_block\_storage() (in module charmhelpers.contrib.storage.linux.ceph), 79

mark\_broker\_action\_done() (in module charmhelpers.contrib.storage.linux.ceph), 79

MemcacheContext (class in charmhelpers.contrib.openstack.context), 55

metadata() (in module charmhelpers.core.hookenv), 22

meter\_info() (in module charmhelpers.core.hookenv), 22

meter\_status() (in module charmhelpers.core.hookenv), 22

midonet\_ctxt() (charmhelpers.contrib.openstack.context.NeutronContext method), 55

missing\_data(charmhelpers.contrib.openstack.context.OSContextGenerator attribute), 56

mkdir() (in module charmhelpers.core.host), 29

mkfs\_xfs() (in module charmhelpers.contrib.storage.linux.utils), 84

model\_name() (in module charmhelpers.core.hookenv), 22

model\_uuid() (in module charmhelpers.core.hookenv), 22

modulo\_distribution() (in module charmhelpers.core.host), 29

monitor\_key\_delete() (in module charmhelpers.contrib.storage.linux.ceph), 79

monitor\_key\_exists() (in module charmhelpers.contrib.storage.linux.ceph), 79

monitor\_key\_get() (in module charmhelpers.contrib.storage.linux.ceph), 79

monitor\_key\_set() (in module charmhelpers.contrib.storage.linux.ceph), 79

mount() (in module charmhelpers.core.host), 29

mount\_volume() (in module charmhelpers.contrib.charmsupport.volumes), 47

mounts() (in module charmhelpers.cli.host), 90

mounts() (in module charmhelpers.core.host), 30

msg() (charmhelpers.coordinator.BaseCoordinator method), 95

## N

nlkv\_ctxt() (charmhelpers.contrib.openstack.context.NeutronContext method), 56

nagios\_exportdir(charmhelpers.contrib.charmsupport.nrpe.NRPE attribute), 45

nagios\_logdir(charmhelpers.contrib.charmsupport.nrpe.NRPE attribute), 45

name(charmhelpers.core.services.helpers.RelationContext attribute), 41

network\_get() (in module charmhelpers.core.hookenv), 22

network\_get\_primary\_address() (in module charmhelpers.core.hookenv), 22

network\_manager(charmhelpers.contrib.openstack.context.NeutronContext attribute), 56

network\_manager() (in module charmhelpers.contrib.openstack.neutron), 59

NetworkServiceContext (class in charmhelpers.contrib.openstack.context), 55

OSContextGenerator(charmhelpers.contrib.openstack.context.NeutronContext method), 56

neutron\_plugin\_attribute() (in module *charmhelpers.contrib.openstack.neutron*), 59  
 neutron\_plugins() (in module *charmhelpers.contrib.openstack.neutron*), 59  
 neutron\_security\_groups (class in *charmhelpers.contrib.openstack.context.NeutronContext* attribute), 56  
 NeutronAPIContext (class in *charmhelpers.contrib.openstack.context*), 55  
 NeutronContext (class in *charmhelpers.contrib.openstack.context*), 55  
 NeutronPortContext (class in *charmhelpers.contrib.openstack.context*), 56  
 no\_ip\_found\_error\_out() (in module *charmhelpers.contrib.network.ip*), 52  
 no\_output() (*charmhelpers.cli.CommandLine* method), 90  
 NoNetworkBinding, 18  
 NotificationDriverContext (class in *charmhelpers.contrib.openstack.context*), 56  
 NRPE (class in *charmhelpers.contrib.charmsupport.nrpe*), 45  
 nrpe\_confdir (*charmhelpers.contrib.charmsupport.nrpe.NRPE* attribute), 45  
 ns\_query() (in module *charmhelpers.contrib.network.ip*), 52  
 nuage\_ctxt() (*charmhelpers.contrib.openstack.context.NeutronContext* method), 56  
 nvp\_ctxt() (*charmhelpers.contrib.openstack.context.NeutronContext* method), 56

**O**  
 oldest\_peer() (in module *charmhelpers.contrib.hahelpers.cluster*), 49  
 open\_port() (in module *charmhelpers.core.hookenv*), 22  
 open\_ports() (in module *charmhelpers.core.hookenv*), 22  
 opened\_ports() (in module *charmhelpers.core.hookenv*), 22  
 openstack\_upgrade\_available() (in module *charmhelpers.contrib.openstack.utils*), 67  
 ordered() (in module *charmhelpers.contrib.openstack.utils*), 67  
 os\_application\_version\_set() (in module *charmhelpers.contrib.openstack.utils*), 67  
 os\_release() (in module *charmhelpers.contrib.openstack.utils*), 67  
 os\_requires\_version() (in module *charmhelpers.contrib.openstack.utils*), 67  
 os\_workload\_status() (in module *charmhelpers.contrib.openstack.utils*), 67  
 OSConfigException, 60  
 OSConfigFlagContext (class in *charmhelpers.contrib.openstack.context*), 56  
 OSConfigRenderer (class in *charmhelpers.contrib.openstack.templating*), 60  
 OSConfigTemplate (class in *charmhelpers.contrib.openstack.templating*), 62  
 OSContextGenerator (class in *charmhelpers.contrib.openstack.context*), 56  
 OutputFormatter (class in *charmhelpers.cli*), 90  
 ovs\_ctxt() (*charmhelpers.contrib.openstack.context.NeutronContext* method), 56  
 owner() (in module *charmhelpers.core.host*), 30

**P**  
 packages (*charmhelpers.contrib.openstack.context.NeutronContext* attribute), 56  
 parse\_bridge\_mappings() (in module *charmhelpers.contrib.openstack.neutron*), 59  
 parse\_data\_port\_mappings() (in module *charmhelpers.contrib.openstack.neutron*), 59  
 parse\_mappings() (in module *charmhelpers.contrib.openstack.neutron*), 60  
 parse\_urls() (*charmhelpers.fetch.BaseFetchHandler* method), 86  
 parse\_vlan\_range\_mappings() (in module *charmhelpers.contrib.openstack.neutron*), 60  
 path\_hash() (in module *charmhelpers.core.host*), 30  
 pausable\_restart\_on\_change() (in module *charmhelpers.contrib.openstack.utils*), 67  
 pause\_unit() (in module *charmhelpers.contrib.openstack.utils*), 68  
 payload\_register() (in module *charmhelpers.core.hookenv*), 23  
 payload\_status\_set() (in module *charmhelpers.core.hookenv*), 23  
 payload\_unregister() (in module *charmhelpers.core.hookenv*), 23  
 peer\_echo() (in module *charmhelpers.contrib.peerstorage*), 70  
 peer\_ips() (in module *charmhelpers.contrib.hahelpers.cluster*), 49



peer\_relation\_id() (in module *charmhelpers.core.hookenv*), 23

peer\_retrieve() (in module *charmhelpers.contrib.peerstorage*), 70

peer\_retrieve\_by\_prefix() (in module *charmhelpers.contrib.peerstorage*), 70

peer\_store() (in module *charmhelpers.contrib.peerstorage*), 70

peer\_store\_and\_set() (in module *charmhelpers.contrib.peerstorage*), 70

peer\_units() (in module *charmhelpers.contrib.hahelpers.cluster*), 49

pg\_ctxt() (*charmhelpers.contrib.openstack.context.NeutronContext* method), 56

PhyNICMTUContext (class in *charmhelpers.contrib.openstack.context*), 57

place\_data\_on\_block\_device() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

plugin (*charmhelpers.contrib.openstack.context.NeutronContext* attribute), 56

plugins() (in module *charmhelpers.fetch*), 87

Pool (class in *charmhelpers.contrib.storage.linux.ceph*), 75

pool\_exists() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

pool\_set() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

PoolCreationError, 76

port\_has\_listener() (in module *charmhelpers.contrib.network.ip*), 52

port\_to\_br() (in module *charmhelpers.contrib.network.ovs*), 49

PortManagerCallback (class in *charmhelpers.core.services.base*), 40

ports\_contains() (*charmhelpers.core.services.base.PortManagerCallback* method), 40

PostgresqlDBContext (class in *charmhelpers.contrib.openstack.context*), 57

previous (*charmhelpers.core.unitdata.Delta* attribute), 37

previous() (*charmhelpers.core.hookenv.Config* method), 17

principal\_unit() (in module *charmhelpers.core.hookenv*), 23

provide\_data() (*charmhelpers.core.services.base.ServiceManager* method), 39

provide\_data() (*charmhelpers.core.services.helpers.RelationContext* method), 41

pwgen() (in module *charmhelpers.core.host*), 30

py() (*charmhelpers.cli.OutputFormatter* method), 91

## Q

quantum\_plugins() (in module *charmhelpers.contrib.openstack.neutron*), 60

## R

raw() (*charmhelpers.cli.OutputFormatter* method), 91

rbd\_exists() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

reconfigure\_services() (*charmhelpers.core.services.base.ServiceManager* method), 39

Record (class in *charmhelpers.core.unitdata*), 37

register() (*charmhelpers.contrib.openstack.templating.OSConfigRender* method), 61

register() (*charmhelpers.core.hookenv.Hooks* method), 18

register\_action() (*charmhelpers.contrib.ansible.AnsibleHooks* method), 44

related (*charmhelpers.contrib.openstack.context.OSContextGenerator* attribute), 57

related\_units() (in module *charmhelpers.core.hookenv*), 23

relation\_clear() (in module *charmhelpers.core.hookenv*), 23

relation\_for\_unit() (in module *charmhelpers.core.hookenv*), 23

relation\_get() (in module *charmhelpers.contrib.peerstorage*), 71

relation\_get() (in module *charmhelpers.core.hookenv*), 23

relation\_id() (in module *charmhelpers.core.hookenv*), 23

relation\_ids() (in module *charmhelpers.core.hookenv*), 23

relation\_set() (in module *charmhelpers.contrib.peerstorage*), 71

relation\_set() (in module *charmhelpers.core.hookenv*), 23

relation\_to\_interface() (in module *charmhelpers.core.hookenv*), 23

relation\_to\_role\_and\_interface() (in module *charmhelpers.core.hookenv*), 23

relation\_type() (in module *charmhelpers.core.hookenv*), 23

relation\_types() (in module *charmhelpers.core.hookenv*), 23

RelationContext (class in *charmhelpers.core.services.helpers*), 41

- relations() (in module *charmhelpers.core.hookenv*), 23
- relations\_for\_id() (in module *charmhelpers.core.hookenv*), 23
- relations\_of\_type() (in module *charmhelpers.core.hookenv*), 23
- released() (*charmhelpers.coordinator.BaseCoordinator* method), 95
- releid (*charmhelpers.coordinator.BaseCoordinator* attribute), 95
- relname (*charmhelpers.coordinator.BaseCoordinator* attribute), 95
- remote\_restart() (in module *charmhelpers.contrib.openstack.utils*), 68
- remote\_service\_name() (in module *charmhelpers.core.hookenv*), 24
- remote\_unit() (in module *charmhelpers.core.hookenv*), 24
- remove() (*charmhelpers.contrib.charmsupport.nrpe.Check* method), 45
- remove\_alternative() (in module *charmhelpers.contrib.openstack.alternatives*), 52
- remove\_by\_mountpoint() (*charmhelpers.core.fstab.Fstab* class method), 13
- remove\_cache\_tier() (*charmhelpers.contrib.storage.linux.ceph.Pool* method), 76
- remove\_check() (*charmhelpers.contrib.charmsupport.nrpe.NRPE* method), 45
- remove\_entry() (*charmhelpers.core.fstab.Fstab* method), 13
- remove\_erasure\_profile() (in module *charmhelpers.contrib.storage.linux.ceph*), 80
- remove\_lvm\_physical\_volume() (in module *charmhelpers.contrib.storage.linux.lvm*), 83
- remove\_password\_expiry() (in module *charmhelpers.contrib.unison*), 85
- remove\_password\_expiry() (in module *charmhelpers.core.host*), 30
- remove\_pool\_quota() (in module *charmhelpers.contrib.storage.linux.ceph*), 80
- remove\_pool\_snapshot() (in module *charmhelpers.contrib.storage.linux.ceph*), 80
- rename\_pool() (in module *charmhelpers.contrib.storage.linux.ceph*), 80
- render() (*charmhelpers.contrib.openstack.templating.OSConfigRenderer* method), 61
- render() (in module *charmhelpers.contrib.templating.pyformat*), 84
- render() (in module *charmhelpers.core.templating*), 34
- render\_template (in module *charmhelpers.core.services.helpers*), 42
- ReplicatedPool (class in *charmhelpers.contrib.storage.linux.ceph*), 76
- request (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRq* attribute), 74
- request\_id (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRsp* attribute), 75
- request\_timestamp() (*charmhelpers.coordinator.BaseCoordinator* method), 95
- requested() (*charmhelpers.coordinator.BaseCoordinator* method), 95
- requests (*charmhelpers.coordinator.BaseCoordinator* attribute), 95
- require() (*charmhelpers.coordinator.BaseCoordinator* method), 95
- reset\_os\_release() (in module *charmhelpers.contrib.openstack.utils*), 68
- resolve\_network\_cidr() (in module *charmhelpers.contrib.network.ip*), 52
- resolve\_ports() (*charmhelpers.contrib.openstack.context.NeutronPort* method), 56
- resource\_get() (in module *charmhelpers.core.hookenv*), 24
- restart\_on\_change() (in module *charmhelpers.core.host*), 30
- restart\_on\_change\_helper() (in module *charmhelpers.core.host*), 31
- resume\_unit() (in module *charmhelpers.contrib.openstack.utils*), 68
- retrieve\_ca\_cert() (in module *charmhelpers.contrib.hahelpers.apache*), 47
- retry\_on\_exception() (in module *charmhelpers.core.decorators*), 13
- role\_and\_interface\_to\_relations() (in module *charmhelpers.core.hookenv*), 24
- rsync() (in module *charmhelpers.core.host*), 31
- run() (*charmhelpers.cli.CommandLine* method), 90
- run() (*charmhelpers.contrib.charmsupport.nrpe.Check* method), 45
- run\_as\_user() (in module *charmhelpers.contrib.unison*), 85
- ## S
- save() (*charmhelpers.core.hookenv.Config* method), 17
- save\_restore() (*charmhelpers.core.services.base.ServiceManager* method), 40

save\_ready() (*charmhelpers.core.services.base.ServiceManager* method), 40

save\_script\_rc() (in module *charmhelpers.contrib.openstack.utils*), 69

send\_request\_if\_needed() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

Serial (class in *charmhelpers.coordinator*), 95

Serializable (class in *charmhelpers.core.hookenv*), 18

series\_upgrade\_complete() (in module *charmhelpers.contrib.openstack.utils*), 69

series\_upgrade\_prepare() (in module *charmhelpers.contrib.openstack.utils*), 69

service() (in module *charmhelpers.core.host*), 31

service\_name() (in module *charmhelpers.core.hookenv*), 24

service\_namespace (in *charmhelpers.contrib.openstack.context.ApacheSSLContext* attribute), 53

service\_pause() (in module *charmhelpers.core.host*), 31

service\_reload() (in module *charmhelpers.core.host*), 32

service\_restart() (in module *charmhelpers.core.host*), 32

service\_restart() (in module *charmhelpers.core.services.base*), 40

service\_resume() (in module *charmhelpers.core.host*), 32

service\_running() (in module *charmhelpers.core.host*), 32

service\_start() (in module *charmhelpers.core.host*), 33

service\_stop() (in module *charmhelpers.core.host*), 33

service\_stop() (in module *charmhelpers.core.services.base*), 40

service\_template (*charmhelpers.contrib.charmsupport.templates.Check* attribute), 45

ServiceCA (class in *charmhelpers.contrib.ssl.service*), 72

ServiceManager (class in *charmhelpers.core.services.base*), 39

set() (*charmhelpers.core.unitdata.Storage* method), 38

set\_app\_name\_for\_pool() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

set\_manager() (in module *charmhelpers.contrib.network.ovs*), 50

set\_nic\_mtu() (in module *charmhelpers.core.host*), 33

set\_Open\_vSwitch\_column\_value() (in module *charmhelpers.contrib.network.ovs*), 50

Manager() (*charmhelpers.contrib.storage.linux.ceph.CephBrokerRq* method), 74

set\_os\_workload\_status() (in module *charmhelpers.contrib.openstack.utils*), 69

set\_pool\_quota() (in module *charmhelpers.contrib.storage.linux.ceph*), 80

set\_release() (*charmhelpers.contrib.openstack.templating.OSConfig* method), 61

set\_unit\_paused() (in module *charmhelpers.contrib.openstack.utils*), 69

set\_unit\_upgrading() (in module *charmhelpers.contrib.openstack.utils*), 69

setup\_aa\_profile() (*charmhelpers.contrib.openstack.context.AppArmorContext* method), 54

SharedDBContext (class in *charmhelpers.contrib.openstack.context*), 67

shortname\_re (*charmhelpers.contrib.charmsupport.nrpe.Check* attribute), 45

signing\_conf (*charmhelpers.contrib.ssl.service.ServiceCA* attribute), 73

Singleton (class in *charmhelpers.coordinator*), 95

snap\_install() (in module *charmhelpers.fetch.snap*), 88

snap\_install\_requested() (in module *charmhelpers.contrib.openstack.utils*), 70

snap\_refresh() (in module *charmhelpers.fetch.snap*), 88

snap\_remove() (in module *charmhelpers.fetch.snap*), 88

snapshot\_pool() (in module *charmhelpers.contrib.storage.linux.ceph*), 81

sniff\_iface() (in module *charmhelpers.contrib.network.ip*), 52

SourceConfigError, 86

strip\_trailing\_slash() (in module *charmhelpers.fetch.archiveurl*), 88

splituser() (in module *charmhelpers.fetch.archiveurl*), 88

ssh\_authorized\_peers() (in module *charmhelpers.contrib.unison*), 85

status\_get() (in module *charmhelpers.core.hookenv*), 24

status\_set() (in module *charmhelpers.core.hookenv*), 24

stop\_services() (*charmhelpers.core.services.base.ServiceManager* method), 40

Storage (class in *charmhelpers.core.unitdata*), 37

storage\_get() (in module *charmhelpers.core.hookenv*), 24

storage\_list() (in module



- charmhelpers.core.hookenv*), 24
- subcommand() (*charmhelpers.cli.CommandLine method*), 90
- subcommand\_builder() (*charmhelpers.cli.CommandLine method*), 90
- SubordinateConfigContext (*class in charmhelpers.contrib.openstack.context*), 57
- subparsers (*charmhelpers.cli.CommandLine attribute*), 90
- supported\_formats (*charmhelpers.cli.OutputFormatter attribute*), 91
- symlink() (*in module charmhelpers.core.host*), 33
- sync\_db\_with\_multi\_ipv6\_addresses() (*in module charmhelpers.contrib.openstack.utils*), 70
- sync\_path\_to\_host() (*in module charmhelpers.contrib.unison*), 85
- sync\_to\_peer() (*in module charmhelpers.contrib.unison*), 85
- sync\_to\_peers() (*in module charmhelpers.contrib.unison*), 85
- SysctlContext (*class in charmhelpers.contrib.openstack.context*), 58
- SyslogContext (*class in charmhelpers.contrib.openstack.context*), 58
- ## T
- tab() (*charmhelpers.cli.OutputFormatter method*), 91
- template (*in module charmhelpers.core.services.helpers*), 42
- TemplateCallback (*class in charmhelpers.core.services.helpers*), 41
- test\_command() (*charmhelpers.cli.CommandLine method*), 90
- token\_cache\_pkgs() (*in module charmhelpers.contrib.openstack.utils*), 70
- translate\_exc() (*in module charmhelpers.core.hookenv*), 24
- ## U
- uid\_exists() (*in module charmhelpers.core.host*), 33
- umount() (*in module charmhelpers.core.host*), 33
- UnhandledSource, 86
- unit\_doomed() (*in module charmhelpers.core.hookenv*), 24
- unit\_get() (*in module charmhelpers.core.hookenv*), 24
- unit\_info() (*in module charmhelpers.contrib.charmhelpers*), 44
- unit\_private\_ip() (*in module charmhelpers.core.hookenv*), 24
- unit\_public\_ip() (*in module charmhelpers.core.hookenv*), 24
- unmount\_volume() (*in module charmhelpers.contrib.charmsupport.volumes*), 47
- UnregisteredHookError, 18
- unset() (*charmhelpers.core.unitdata.Storage method*), 38
- unsetrange() (*charmhelpers.core.unitdata.Storage method*), 38
- update() (*charmhelpers.core.unitdata.Storage method*), 38
- update\_json\_file() (*in module charmhelpers.contrib.openstack.utils*), 70
- update\_machine\_state() (*in module charmhelpers.contrib.saltstack*), 72
- update\_pool() (*in module charmhelpers.contrib.storage.linux.ceph*), 81
- update\_relations() (*in module charmhelpers.contrib.templating.contexts*), 84
- updatedb() (*in module charmhelpers.core.host*), 34
- user (*charmhelpers.contrib.openstack.context.ApacheSSLContext attribute*), 53
- user\_exists() (*in module charmhelpers.core.host*), 34
- ## V
- valid\_hacluster\_config() (*in module charmhelpers.contrib.hahelpers.cluster*), 49
- valid\_snap\_channel() (*in module charmhelpers.fetch.snap*), 89
- validator() (*in module charmhelpers.contrib.storage.linux.ceph*), 81
- VersionsContext (*class in charmhelpers.contrib.openstack.context*), 58
- VolumeAPIContext (*class in charmhelpers.contrib.openstack.context*), 58
- VolumeConfigurationError, 46
- ## W
- wait\_for\_machine() (*in module charmhelpers.contrib.charmhelpers*), 44
- wait\_for\_page\_contents() (*in module charmhelpers.contrib.charmhelpers*), 44

`wait_for_relation()` (in module `charmhelpers.contrib.charmhelpers`), 44

`wait_for_unit()` (in module `charmhelpers.contrib.charmhelpers`), 44

`was_ready()` (`charmhelpers.core.services.base.ServiceManager` method), 40

`WorkerConfigContext` (class in `charmhelpers.contrib.openstack.context`), 58

`workload_state_compare()` (in module `charmhelpers.contrib.openstack.utils`), 70

`write()` (`charmhelpers.contrib.charmsupport.nrpe.Check` method), 45

`write()` (`charmhelpers.contrib.charmsupport.nrpe.NRPE` method), 45

`write()` (`charmhelpers.contrib.openstack.templating.OSConfigRenderer` method), 61

`write_all()` (`charmhelpers.contrib.openstack.templating.OSConfigRenderer` method), 62

`write_authorized_keys()` (in module `charmhelpers.contrib.unison`), 86

`write_file()` (in module `charmhelpers.core.host`), 34

`write_known_hosts()` (in module `charmhelpers.contrib.unison`), 86

`write_service_config()` (`charmhelpers.contrib.charmsupport.nrpe.Check` method), 45

`WSGIWorkerConfigContext` (class in `charmhelpers.contrib.openstack.context`), 58

## Y

`yaml()` (`charmhelpers.cli.OutputFormatter` method), 91

`yaml()` (`charmhelpers.core.hookenv.Serializable` method), 18

## Z

`zap_disk()` (in module `charmhelpers.contrib.storage.linux.utils`), 84

`ZeroMQContext` (class in `charmhelpers.contrib.openstack.context`), 58