Kokki Documentation

Release 0.4.1

Samuel Stauffer

July 20, 2016

Kokki is a configuration management framework inspired by Chef.

Installation

Stable releases can be installed using:

```
easy_install kokki
```

or:

```
pip install kokki
```

Source

You can find the latest version at http://github.com/samuel/kokki

Overview

A `Kitchen` to refer to a collection of Kokki configuration files and associated cookbooks, recipes, providers, and resources.

Todo

Make a command line utility/option to set up a new Kitchen with a reasonable default setup including pointing at the main Kokki cookbook collection and proper directory structure.

A kokki configuration is composed of a few pieces, kept in a `Kitchen`:

configuration file

> specifies cookbook paths and roles. Conventionally this is named `config.yaml` and put in the top level directory of the `Kitchen`.

provider

> os/platform specific code to bring the system to the state given by a resource. Most of the ones you'll need are provided by the default Kokki installation and can be found in `kokki/providers`.

cookbooks

> a collection of recipes and extra resources and providers. A pre-written collection of recipes is available at http://github.com/samuel/kokki-cookbooks

recipe

> a script that includes a various resources describing the expected state of a system. These are usually kept within a cookbook.

resource

> describes a piece of the system configuration state (e.g. described a file, a user, etc..). The default Resources are in `kokki/resources`.
>
> Recipe specific resources are described within the `resources` directory within a recipe.

Todo

how to add global resources to Kokki so they can be used by multiple recipes without modifying kokki core? May require new conf.yaml item?

Operation

Todo

finish this section

The command line is:

```
kokki config_file role
```

Order of operation is:

```
Read config_file -- this is a .yaml file
Read directories in `cookbook_paths:` item in config_file.yaml
For dir in `cookbook_paths`:
    import package, run __init__.py as usual
```

Example

config.py:

```python
import os.path

COOKBOOK_PATH = os.path.join(os.path.dirname(os.path.dirname(os.path.abspath(__file__))), "cookbooks'

def base(kit):
    kit.add_cookbook_path(COOKBOOK_PATH, "kokki.cookbooks")
    kit.update_config({
        "limits": [
            dict(domain="mongodb", type="soft", item="nofile", value="10000"),
            dict(domain="mongodb", type="hard", item="nofile", value="10000"),
            dict(domain="nginx", type="soft", item="nofile", value="10000"),
            dict(domain="nginx", type="hard", item="nofile", value="10000"),
            dict(domain="www-data", type="soft", item="nofile", value="10000"),
            dict(domain="www-data", type="hard", item="nofile", value="10000"),
            dict(domain="root", type="soft", item="nofile", value="30000"),
            dict(domain="root", type="hard", item="nofile", value="30000"),
        ],
    })
    kit.include_recipe("users", "sudo", "limits")

def web(kit):
    base(kit)
    kit.update_config({
        "example.web_port": 8080,
    })
    kit.include_recipe("example")
```

cookbooks/example/__init__.py:

```python
# [empty]
```

cookbooks/example/metadata.py:

```python
__description__ = "Example cookbook"
__config__ = {
    "foo.content": dict(
        description = "Content of the file /tmp/foo",
        default = "This is the default text",
    ),
}
```

cookbooks/example/recipes/default.py:

```python
from kokki import *

Package("git-core")
```

cookbooks/example/recipes/web.py:

```python
from kokki import *

Package("apache2")
File("/etc/apache2/ports.conf",
    owner = "www-data",
    content = "Listen %d\n" % env.example.web_port)
```

To run the 'web' role on the local system:

```
kokki config.py web

or

python -m kokki.runner config.py web
```

TOC

6.1 Resource Types

6.1.1 File

action create(default), delete, touch

path String: Path to the file (defaults to 'name')

backup Boolean: Create a backup of any replaced files (unimplemented)

mode Integer: Numerical mode for the file

owner String/Integer: UID or username

group String/Integer: GID or groupname

content Source: Template or string

6.1.2 Directory

action create(default), delete

path String: Path to the directory (defaults to 'name')

mode Integer: Numerical mode for the file

owner String/Integer: UID or username

group String/Integer: GID or groupname

recursive Boolean: Recursively create or delete the folder and it's parents (default False)

6.1.3 Link

action create(default), delete

path String: Path to the link (defaults to 'name')

to String: Path to the file or directory to link to

hard Boolean: Create a hard link (default False)

6.1.4 Execute

action run(default)

command String: Command to execute (defaults to 'name')

creates String: Path to a file or directory that is created by the command. If the path exists then don't execute the command.

cwd String: Working directory when executing the command

environment: Dict: Extra environment variables

user: String/Integer: UID or username to run the command as (unimplemented)

group: String/Integer: GID or groupname to run the command as (unimplemented)

returns: Integer: Expected return value for success (default 0)

timeout: Integer: Max number of seconds the command is allowed to execute for (unimplemented)

6.1.5 Script

action run(default)

code String: Shell script

cwd String: Working directory when executing the script

interpreter String: Interpreter to run the script with (default /bin/bash)

6.1.6 Mount

action mount(default), umount, remount, enable, disable

mount_point String: Path to the mount point (defaults to 'name')

device String: Device to mount

fstype String: Filesystem type

options List: List of options given to mount (default ["defaults"])

dump Integer: dump value in fstab (default 0)

passno Integer: passno value in fstab (default 2)

6.1.7 Package

action install(default), upgrade, remove, purge

package_name String: Name of package (defaults to 'name')

version String: Version of package to install