pastry Documentation

Stephen Breyer-Menke

Mar 09, 2018

Table of Contents

CHAPTER 1

PastryClient

1.1 Supported Operations

Initialize	Initialize the client with the chef server info
Load Config	Load server config from a yaml file
Get Url	Get the server and endpoint for the request
Call	Call a resource on the chef server

1.2 API Reference

class pastry.pastry_client.**PastryClient**

PastryClient is used by the resources to send requests to chef. You can use the PastryClient to initialize the config to use for the chef server.

classmethod call(*endpoint*, *method='GET'*, *data=None*)

Send an http request to the chef server api

Parameters

- **endpoint** (*string*) – The endpoint for the resource being called
- **method** (*string*) – The HTTP method
- **data** (*hash*) – The body of the request

Returns The json response from the server

Return type hash

classmethod get_url(*endpoint*)

Fetches the server and updates the endpoint with the organization

Parameters **endpoint** (*string*) – The endpoint being called on the chef server

Returns The server url and updated endpoint

Return type tuple

classmethod initialize(*server*, *organization*, *client*, *keypath*, *verify*)
Initializes the server connection info

Parameters

- **server** (*string*) – The url for the chef server e.g. https://my.chef.server
- **organization** (*string*) – The name of the cheff org to use
- **client** (*string*) – The client/username to use
- **keypath** (*string*) – The path to the pem for the client/user
- **verify** (*boolean*) – Verify the ssl cert on requests

classmethod load_config(*config_path=None*)
Load server config from a yaml file

If no config_path is specified it will try to load config from `$HOME/.chef/pastry.yaml`

Note: This method is automatically called if an http requests is made and the client has not yet been initialised

---

Parameters **config_path** (*string*) – The path to the config file on the local filesystem

CHAPTER 2

Search

The search resource provides access to chef search. It supports filtering out only required fields instead of returning the entire objects from the chef server.

2.1 Supported Operations

Index	Fetch a list of available search indexes
Run	Run a search against an index on the chef server

2.2 API Reference

class pastry.resources.search.**Search**

Provides access to chef *search*

classmethod index()

Fetches the available search indexes

Returns The search indexes

Return type hash

classmethod run(*index*, *query='*:*'*, *rows=1000*, *start=0*, *filters=None*)

Runs the search query against the index. The default will return the a list of all of the properties of the match. Use the filters to select which fields should be returned by the query.

The filters should be a hash where the key is the name of the field in the returned hash, and the value is a space separated list of where in the index to find the value. e.g.:

```
{
    'name': [ 'name' ],
    'ip': [ 'ipaddress' ],
```

(continues on next page)

(continued from previous page)

```
    'kernel_version': [ 'kernel', 'version' ]
}
```

Parameters

- **index** (*string*) – The index to search against
- **query** (*string*) – The SOLR query to match the index against
- **rows** (*integer*) – The maximum number of rows to return
- **start** (*integer*) – The row to start at
- **filters** (*hash*) – The filters to apply to the result

Returns The matching (and filtered) results

Return type hash

CHAPTER 3

Users

Wraps the chef user account. In order to modify users in chef you will need server admin privledges. Server admins are supported in chef 12.4.1 and above. If you are using an older version of chef server you will have to use the pivotal user (not recommended) to be able to modify users. More info on chef server admins can be found in the chef server admins documentation.

3.1 Supported Operations

Index	Fetch a list of all of the users
Get	Lookup user info
Create	Create a new user
Update	Update a group
Delete	Delete a group
Exists	Check if a user exists
Invite	Invite a user to an org

3.2 API Reference

class pastry.resources.users.**Users**

Provides access to the chef *users* resource

classmethod create(*user*)

Creates a new chef user on the server

The user hash should be in the form:

```
{
    'username': <username>,
    'display_name': <display_name>,
```

(continues on next page)

(continued from previous page)

```
    'first_name': <first name>,
    'middle_name': <middle name>, # optional
    'last_name': <last name>,
    'email': <email>,
    'password': <password>
}
```

Parameters **user** (*hash*) – The user to create

Returns The username and url

Return type hash

classmethod delete(*username*)
Deletes a user from the chef server

Parameters **username** – The User's username

Returns The deleted user's name

Return type hash

classmethod exists(*username*)
Checks if a user exists on the chef server

Parameters **username** – The User's username

Returns If the user exists

Return type boolean

classmethod get(*username*)
Fetches a user from the chef server

Parameters **username** (*string*) – The User's username

Returns The chef user

Return type hash

classmethod index()
Fetches all of the users on the chef server

Returns All the chef users with a url for each user

Return type hash

classmethod invite(*username*, *orgname*)
Invite a user to an org

Parameters

- **username** (*string*) – The User's username
- **orgname** (*string*) – The chef organization

Returns If the request was successful

Return type boolean

classmethod update(*username*, *user*)
Updates a user on the chef server

Parameters

- **username** (*string*) – The User's username
- **user** (*hash*) – The User members and content

Returns The username and url

Return type hash

CHAPTER 4

Groups

The groups resource adds support for managing chef groups.

4.1 Supported Operations

Index	Fetch a list of all of the groups
Get	Lookup group info
Create	Create a new group
Update	Update a group
Delete	Delete a group
Exists	Check if a group exists

4.2 API Reference

class pastry.resources.groups.**Groups**

Provides access to the chef *groups* resource

classmethod create(*group*)

Creates a new chef group on the server

The group hash should be in the form:

```
{
    'groupname': <groupname>,
    'name': <groupname>,
    'actors': { #optional
        'users': <list of usernames>, # optional
        'clients': <list of clients>, #optional
        'groups': <list of groups> #optional
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Parameters **group** (*hash*) – The group to create

Returns The groupname and url

Return type hash

classmethod delete(*groupname*)
Deletes a group from the chef server

Parameters **groupname** (*string*) – The Group's groupname

Returns The deleted group's name

Return type hash

classmethod exists(*groupname*)
Checks if a group exists on the chef server

Parameters **username** – The Group's groupname

Returns If the group exists

Return type boolean

classmethod get(*groupname*)
Fetches a group from the chef server

Note: Chef returns a slightly different format to what it expects for create/update groups.

Parameters **groupname** (*string*) – The Group's name

Returns The chef group and members

Return type hash

classmethod index()
Fetches all of the groups on the chef server

Returns All the chef groups with a url for each group

Return type hash

classmethod update(*groupname*, *group*)
Updates a group on the chef server

the group hash should be in the same format as for create

Parameters

- **groupname** (*string*) – The Group's groupname
- **group** (*hash*) – The Group members and content

Returns The groupname and url

Return type hash

CHAPTER 5

Nodes

The groups module adds support for chef nodes.

5.1 Supported Operations

Exists	Check if a node exists
Get Acl	Fetch a node's acl
Set Permission	Set a permission for a node

5.2 API Reference

class pastry.resources.nodes.**Nodes**
Provides access to the chef nodes

classmethod exists(*nodename*)
Checks if a node exists on the chef server

Parameters **nodename** (*string*) – The Node's nodename

Returns If the node exists

Return type boolean

classmethod get_acl(*nodename*)
Gets the access control list for the node

Parameters **nodename** (*string*) – The Node's nodename

Returns The acl for the node

Return type hash

classmethod set_permission(*nodename*, *permission*, *actors*)

Grants the specified actors a permission on the node. Chef only supports setting one permission at a time.

The actors hash should be in the form:

```
{
    'actors': <list of usernames>
    'groups': <list of groupnames>
}
```

Parameters

- **nodename** (*string*) – The Node's nodename
- **permission** (*string*) – One of: create, read, update, delete, grant
- **actors** (*hash*) – The set of actors to grant this permission to

Returns empty hash

Type hash

CHAPTER 6

Environments

The environments resource adds support for managing chef environments.

6.1 Supported Operations

Index	Fetch a list of all of the users
Get	Lookup user info
Create	Create a new environment
Update	Update an environment
Delete	Delete an environment
Exists	Check if an environment exists

6.2 API Reference

CHAPTER 7

Cookbooks

The cookbooks resource adds support for querying chef cookbooks.

7.1 Supported Operations

Index	Fetch a list of all of the groups
Exists	Check if a group exists
Contents	Get the list of files in a cookbook
Parse Filename	Splits up the path to a file in a cookbook
File Content	Read the contents of a file

7.2 API Reference

class pastry.resources.cookbooks.**Cookbooks**
Provides methods for interacting with chef cookbooks

classmethod contents(*cookbook*, *version='_latest'*)
Fetches the cookbooks list of files that chef knows about

Parameters

- **cookbook** (*string*) – The cookbook's name
- **version** (*string*) – The cookbook's version

Returns All of the files the cookbook knows about

Return type hash

classmethod exists(*cookbook*)
Checks if a cookbook exists on the chef server

Parameters **cookbook** (*string*) – The Cookbook's name

Returns If the cookbook exists

Return type boolean

classmethod file_content(*cookbook*, *filename*, *version='_latest'*)
Fetches the contents of a specific file in a cookbook

Parameters

- **cookbook** (*string*) – The cookbook's name
- **filename** (*string*) – The name (and path) of the file to fetch
- **version** (*string*) – The cookbook's version

Returns The raw contents of the specified file

Return type string

classmethod index()
Fetches all of the cookbooks (and versions) on the chef server

Returns All the chef cookbooks and versions

Return type hash

classmethod parse_filename(*filename*)
Splits the file path so that it can be used to call the chef api

Parameters **filename** (*string*) – The file's path relative to the cookbook root

Returns The type of file, specificity, and filename

Return type iterable

CHAPTER 8

Pushy

Provides a wrapper to the chef pushy endpoints. node_states (status) is the only supported endpoint at this time.

8.1 Supported Operations

Status	Check the push jobs status on a node

8.2 API Reference

class pastry.resources.pushy.**Pushy**

Provides access to the chef pushy api

classmethod status(*nodename*)

Checks the push jobs status of a node

Parameters **nodename** (*string*) – The Node's nodename

Returns If pushy jobs is available on the node

Return type boolean

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index