

---

# **Bitbucket-API Documentation**

*Release 0.4.4dev*

**Baptiste Millou**

**Nov 20, 2017**



---

# Contents

---

<b>1</b>	<b>Installation</b>	<b>1</b>
1.1	Pip	1
1.2	Get the Code & contribute	1
1.3	Test	1
<b>2</b>	<b>Usage</b>	<b>3</b>
2.1	Public	3
2.2	Private	3
2.3	Examples	3
<b>3</b>	<b>bitbucket Package</b>	<b>5</b>
3.1	bitbucket Package	5
3.2	Bitbucket Module	6
3.3	issue Module	7
3.4	issue_comment Module	8
3.5	repository Module	8
3.6	service Module	9
3.7	ssh Module	9
3.8	Subpackages	10
<b>4</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



### 1.1 Pip

Installing Bitbucket-API is simple with pip:

```
pip install Bitbucket-API
```

### 1.2 Get the Code & contribute

Bitbucket-API is hosted on GitHub, where the code is always available.

You can either clone the public repository:

```
git clone git@github.com:Sheeprider/BitBucket-api.git
```

Download the tarball:

```
curl -OL https://github.com/Sheeprider/BitBucket-api/tarball/master
```

Or, download the zipball:

```
curl -OL https://github.com/Sheeprider/Bitbucket-API/zipball/master
```

### 1.3 Test

Run public tests:

```
site-packages$> python -m bitbucket.tests.public
```

Run private tests. Require **USERNAME** and **PASSWORD** or **USERNAME**, **CONSUMER\_KEY** and **CONSUMER\_SECRET** in *bitbucket/tests/private/settings.py*:

```
site-packages$> python -m bitbucket.tests.private
```

## 2.1 Public

You can access any public repository on Bitbucket, but some actions won't be available without credentials.

```
>>> from bitbucket.bitbucket import Bitbucket
>>> bb = Bitbucket(USERNAME, repo_name_or_slug='public_slug')
>>> success, result = bb.repository.delete()
>>> print success
False
```

## 2.2 Private

With the correct credentials you can access private repositories on Bitbucket.

```
>>> from bitbucket.bitbucket import Bitbucket
>>> bb = Bitbucket(USERNAME, PASSWORD, 'private_slug')
>>> success, result = bb.repository.get()
>>> print success, result
True {...}
```

## 2.3 Examples

Connect using OAuth

```
>>> import webbrowser
>>> from bitbucket.bitbucket import Bitbucket
>>> bb = Bitbucket(USERNAME)
>>> # First time we need to open up a browser to enter the verifier
>>> if not OAUTH_ACCESS_TOKEN and not OAUTH_ACCESS_TOKEN_SECRET:
```

```
>>> bb.authorize(CONSUMER_KEY, CONSUMER_SECRET, 'http://localhost/')
>>> # open a webbrowser and get the token
>>> webbrowser.open(bb.url('AUTHENTICATE', token=bb.access_token))
>>> # Copy the verifier field from the URL in the browser into the console
>>> oauth_verifier = raw_input('Enter verifier from url [oauth_verifier]')
>>> bb.verify(oauth_verifier)
>>> OAUTH_ACCESS_TOKEN = bb.access_token
>>> OAUTH_ACCESS_TOKEN_SECRET = bb.access_token_secret
>>> else:
>>> bb.authorize(CONSUMER_KEY, CONSUMER_SECRET, 'http://localhost/', OAUTH_ACCESS_
↵TOKEN, OAUTH_ACCESS_TOKEN_SECRET)
```

List all repositories for a user (from @matthew-campbell):

```
>>> from bitbucket.bitbucket import Bitbucket
>>> bb = Bitbucket(USERNAME, PASSWORD)
>>> success, repositories = bb.repository.all()
>>> for repo in sorted(repositories):
>>>     p = '+'
>>>     if repo['is_private']:
>>>         p = '-'
>>>     print('{{}} {}, {}, {}'.format(p, repo['name'], repo['last_updated'], repo['scm
↵']))
>>> print('Total {}'.format(len(repositories)))
```



### 3.1 bitbucket Package

Bitbucket has a REST API publicly available, this package provide methods to interact with it. It allows you to access repositories and perform various actions on them.

Various usages :

```
from bitbucket.bitbucket import Bitbucket

# Access a public repository
bb = Bitbucket(USERNAME, repo_name_or_slug="public_repository")

# Access a private repository
bb = Bitbucket(USERNAME, PASSWORD, repo_name_or_slug="private_repository")

# Access a private repository through oauth
bb = Bitbucket(USERNAME, repo_name_or_slug="public_repository")
bb.authorize(CONSUMER_KEY, CONSUMER_SECRET, 'http://localhost/')

# Access your working repository
success, result = bb.repository.get()

# Create a repository, and define it as your working repository
success, result = bb.repository.create("repository_slug")
bb.repo_slug = "repository_slug"

# Update your working repository
success, result = bb.repository.update(description='new description')

# Delete a repository
success, result = bb.repository.delete("repository_slug")

# Download a repository as an archive
```

```
success, archive_path = bb.repository.archive()

# Access user informations
success, result = bb.get_user(username=USERNAME)

# Access tags and branches
success, result = bb.get_tags()
success, result = bb.get_branches()

# Access, create, update or delete a service (hook)
success, result = bb.service.get(service_id=SERVICE_ID)
success, result = bb.service.create(service=u'POST', URL='http://httpbin.org/')
success, result = bb.service.update(service_id=SERVICE_ID, URL='http://google.com')
success, result = bb.service.delete(service_id=SERVICE_ID)

# Access, create or delete an SSH key
success, result = bb.ssh.get(key_id=SSH_ID)
success, result = bb.ssh.create(key=r'ssh-rsa a1b2c3d4e5', label=u'my key')
success, result = bb.ssh.delete(key_id=SSH_ID)

# Access, create, update or delete an issue
success, result = bb.issue.get(issue_id=ISSUE_ID)
success, result = bb.issue.create(
    title=u'Issue title',
    content=u'Issue content',
    responsible=bb.username,
    status=u'new',
    kind=u'bug')
success, result = bb.issue.update(issue_id=ISSUE_ID, content='New content')
success, result = bb.issue.delete(issue_id=ISSUE_ID)

# Access, create, update or delete an issue comment
success, result = bb.issue.comment.get(comment_id=COMMENT_ID)
success, result = bb.issue.comment.create(content='Content')
success, result = bb.issue.comment.update(
    comment_id=COMMENT_ID,
    content='New content')
success, result = bb.issue.comment.delete(comment_id=COMMENT_ID)
```

## 3.2 Bitbucket Module

**class** bitbucket.bitbucket.**Bitbucket** (*username='', password='', repo\_name\_or\_slug=''*)

This class lets you interact with the bitbucket public API.

### **auth**

Return credentials for current Bitbucket user.

**authorize** (*consumer\_key, consumer\_secret, callback\_url=None, access\_token=None, access\_token\_secret=None*)

Call this with your consumer key, secret and callback URL, to generate a token for verification.

**dispatch** (*method, url, auth=None, params=None, \*\*kwargs*)

Send HTTP request, with given method, credentials and data to the given URL, and return the success and the result on success.

**finalize\_oauth** (*access\_token, access\_token\_secret*)

Called internally once auth process is complete.

**get\_branches** (*repo\_slug=None*)

Get a single repository on Bitbucket and return its branches.

**get\_privileges** ()

Get privileges for this user.

**get\_tags** (*repo\_slug=None*)

Get a single repository on Bitbucket and return its tags.

**get\_user** (*username=None*)

Returns user informations. If username is not defined, tries to return own informations.

**password**

Return your repository's password.

**repo\_slug**

Return your repository's slug name.

**url** (*action, \*\*kwargs*)

Construct and return the URL for a specific API service.

**username**

Return your repository's username.

**verify** (*verifier, consumer\_key=None, consumer\_secret=None, access\_token=None, access\_token\_secret=None*)

After converting the token into verifier, call this to finalize the authorization.

### 3.3 issue Module

**class** `bitbucket.issue.Issue` (*bitbucket, issue\_id=None*)

This class provide issue-related methods to Bitbucket objects.

**all** (*repo\_slug=None, params=None*)

Get issues from one of your repositories.

**create** (*repo\_slug=None, \*\*kwargs*)

Add an issue to one of your repositories. Each issue require a different set of attributes, you can pass them as keyword arguments (`attributename='attributevalue'`). Attributes are:

- **title**: The title of the new issue.
- **content**: The content of the new issue.
- **component**: The component associated with the issue.
- **milestone**: The milestone associated with the issue.
- **version**: The version associated with the issue.
- **responsible**: The username of the person responsible for the issue.
- **status**: The status of the issue (new, open, resolved, on hold, invalid, duplicate, or wontfix).
- **kind**: The kind of issue (bug, enhancement, or proposal).

**delete** (*issue\_id, repo\_slug=None*)

Delete an issue from one of your repositories.

**get** (*issue\_id, repo\_slug=None*)

Get an issue from one of your repositories.

**issue\_id**

Your repository slug name.

**update** (*issue\_id*, *repo\_slug=None*, *\*\*kwargs*)

Update an issue to one of your repositories. Each issue require a different set of attributes, you can pass them as keyword arguments (attributename='attributevalue'). Attributes are:

- title: The title of the new issue.
- content: The content of the new issue.
- component: The component associated with the issue.
- milestone: The milestone associated with the issue.
- version: The version associated with the issue.
- responsible: The username of the person responsible for the issue.
- status: The status of the issue (new, open, resolved, on hold, invalid, duplicate, or wontfix).
- kind: The kind of issue (bug, enhancement, or proposal).

## 3.4 issue\_comment Module

**class** `bitbucket.issue_comment.IssueComment` (*issue*)

This class provide issue's comments related methods to Bitbucket objects.

**all** (*issue\_id=None*, *repo\_slug=None*)

Get issue comments from one of your repositories.

**create** (*issue\_id=None*, *repo\_slug=None*, *\*\*kwargs*)

Add an issue comment to one of your repositories. Each issue comment require only the content data field the system autopopulate the rest.

**delete** (*comment\_id*, *issue\_id=None*, *repo\_slug=None*)

Delete an issue from one of your repositories.

**get** (*comment\_id*, *issue\_id=None*, *repo\_slug=None*)

Get an issue from one of your repositories.

**update** (*comment\_id*, *issue\_id=None*, *repo\_slug=None*, *\*\*kwargs*)

Update an issue comment in one of your repositories. Each issue comment require only the content data field the system autopopulate the rest.

## 3.5 repository Module

**class** `bitbucket.repository.Repository` (*bitbucket*)

This class provide repository-related methods to Bitbucket objects.

**all** ()

Return own repositories.

**archive** (*repo\_slug=None*, *format='zip'*, *prefix=''*)

Get one of your repositories and compress it as an archive. Return the path of the archive.

format parameter is curenly not supported.

**create** (*repo\_name*, *scm='git'*, *private=True*, *\*\*kwargs*)

Creates a new repository on own Bitbucket account and return it.

**delete** (*repo\_slug=None*)

Delete a repository on own Bitbucket account. Please use with caution as there is NO confirmation and NO undo.

**get** (*repo\_slug=None*)

Get a single repository on Bitbucket and return it.

**public** (*username=None*)

Returns all public repositories from an user. If username is not defined, tries to return own public repos.

**update** (*repo\_slug=None, \*\*kwargs*)

Updates repository on own Bitbucket account and return it.

## 3.6 service Module

**class** `bitbucket.service.Service` (*bitbucket*)

This class provide services-related methods to Bitbucket objects.

**all** (*repo\_slug=None*)

Get all services (hook) from one of your repositories.

**create** (*service, repo\_slug=None, \*\*kwargs*)

Add a service (hook) to one of your repositories. Each type of service require a different set of additionnal fields, you can pass them as keyword arguments (`fieldname='fieldvalue'`).

**delete** (*service\_id, repo\_slug=None*)

Delete a service (hook) from one of your repositories. Please use with caution as there is NO confirmation and NO undo.

**get** (*service\_id, repo\_slug=None*)

Get a service (hook) from one of your repositories.

**update** (*service\_id, repo\_slug=None, \*\*kwargs*)

Update a service (hook) from one of your repositories.

## 3.7 ssh Module

**class** `bitbucket.ssh.SSH` (*bitbucket*)

This class provide ssh-related methods to Bitbucket objects.

**all** ()

Get all ssh keys associated with your account.

**create** (*key=None, label=None*)

Associate an ssh key with your account and return it.

**delete** (*key\_id=None*)

Delete one of the ssh keys associated with your account. Please use with caution as there is NO confirmation and NO undo.

**get** (*key\_id=None*)

Get one of the ssh keys associated with your account.

## 3.8 Subpackages

### 3.8.1 bitbucket.tests Package

#### tests Package

##### public Module

```
class bitbucket.tests.public.AnonymousBitbucketTest (methodName='runTest')
    Bases: unittest.case.TestCase
    Bitbucket test base class.
    setUp()
        Create a new anonymous Bitbucket...
    tearDown()
        Destroy the Bitbucket...

class bitbucket.tests.public.BitbucketAnonymousMethodsTest (methodName='runTest')
    Bases: bitbucket.tests.public.AnonymousBitbucketTest
    Test Bitbucket anonymous methods.
    test_get_none_public_repos()
        Test public_repos on specific user.
    test_get_none_user()
        Test get_user with no username.
    test_get_public_repos()
        Test public_repos on specific user.
    test_get_self_public_repos()
        Test public_repos on specific user.
    test_get_self_user()
        Test get_user on self username.
    test_get_user()
        Test get_user on specific user.

class bitbucket.tests.public.BitbucketUtilitiesTest (methodName='runTest')
    Bases: bitbucket.tests.public.AnonymousBitbucketTest
    Test Bitbucket utilities functions.
    test_auth()
    test_default_credential()
    test_dispatch_delete()
    test_dispatch_get()
    test_dispatch_post()
    test_dispatch_put()
    test_password()
    test_repo_slug()
    test_url_complex
```

```
test_url_simple()
```

```
test_username()
```

## Subpackages

### bitbucket.tests.private Package

#### private Package

#### issue Module

```
class bitbucket.tests.private.issue.IssueAuthenticatedMethodsTest (methodName='runTest')
```

Bases: *bitbucket.tests.private.private.AuthenticatedBitbucketTest*

Testing bitbucket.issue methods.

```
test_CRUD()
```

Test issue create/read/update/delete.

```
test_all()
```

Test get all issues.

#### issue\_comment Module

```
class bitbucket.tests.private.issue_comment.IssueCommentAuthenticatedMethodsTest (methodName='runTest')
```

Bases: *bitbucket.tests.private.private.AuthenticatedBitbucketTest*

Testing bitbucket.issue.comments methods.

```
setUp()
```

Add an issue to the test repository and save it's id.

```
tearDown()
```

Delete the issue.

```
test_CRUD()
```

Test issue comment create/read/update/delete.

```
test_all()
```

Test get all issue comments.

#### private Module

```
class bitbucket.tests.private.private.AuthenticatedBitbucketTest (methodName='runTest')
```

Bases: *unittest.case.TestCase*

Bitbucket test base class for authenticated methods.

```
setUp()
```

Creating a new authenticated Bitbucket...

```
tearDown()
```

Destroying the Bitbucket...

```
class bitbucket.tests.private.private.BitbucketAuthenticatedMethodsTest (methodName='runTest')
    Bases: bitbucket.tests.private.private.AuthenticatedBitbucketTest
```

Testing Bitbucket anonymous methods.

```
test_get_branches ()
    Test get_branches.
```

```
test_get_tags ()
    Test get_tags.
```

## repository Module

```
class bitbucket.tests.private.repository.ArchiveRepositoryAuthenticatedMethodsTest (methodName='runTest')
    Bases: bitbucket.tests.private.private.AuthenticatedBitbucketTest
```

Testing bitbucket.repository.archive method, which require custom setUp and tearDown methods.

test\_archive require a commit to download the repository.

```
setUp ()
    Clone the test repo locally, then add and push a commit.
```

```
tearDown ()
    Delete the git folder.
```

```
test_archive ()
    Test repository download as archive.
```

```
class bitbucket.tests.private.repository.RepositoryAuthenticatedMethodsTest (methodName='runTest')
    Bases: bitbucket.tests.private.private.AuthenticatedBitbucketTest
```

Testing bitbucket.repository methods.

```
test_all ()
    Test get all repositories.
```

```
test_create ()
    Test repository creation.
```

```
test_delete ()
    Test repository deletion.
```

```
test_get ()
    Test get a repository.
```

```
test_update ()
    Test repository update.
```

```
bitbucket.tests.private.repository.skipUnlessHasGit (f)
    This decorator pass the test if git is not found.
```

## service Module

```
class bitbucket.tests.private.service.ServiceAuthenticatedMethodsTest (methodName='runTest')
    Bases: bitbucket.tests.private.private.AuthenticatedBitbucketTest
```

Testing bitbucket.service methods.

```
test_CRUD ()
    Test service create/read/update/delete.
```



```
test_all ()  
    Test get all services.
```

### settings Module

Set **USERNAME** and **PASSWORD** or **USERNAME**, **CONSUMER\_KEY** and **CONSUMER\_SECRET** in *bitbucket/tests/private/settings.py* if you want to run tests for private methods.

### ssh Module

```
class bitbucket.tests.private.ssh.SSHAuthenticatedMethodsTest (methodName='runTest')  
    Bases: bitbucket.tests.private.private.AuthenticatedBitbucketTest  
  
    Testing bitbucket.ssh methods.  
  
    test_CRUD ()  
        Test ssh create/read/delete.  
  
    test_all ()  
        Test get all sshs.
```



## CHAPTER 4

---

### Indices and tables

---

- genindex
- modindex
- search



### b

- bitbucket, 5
- bitbucket.bitbucket, 6
- bitbucket.issue, 7
- bitbucket.issue\_comment, 8
- bitbucket.repository, 8
- bitbucket.service, 9
- bitbucket.ssh, 9
- bitbucket.tests, 10
- bitbucket.tests.private, 11
- bitbucket.tests.private.issue, 11
- bitbucket.tests.private.issue\_comment,  
11
- bitbucket.tests.private.private, 11
- bitbucket.tests.private.repository, 12
- bitbucket.tests.private.service, 12
- bitbucket.tests.private.ssh, 13
- bitbucket.tests.public, 10



**A**

all() (bitbucket.issue.Issue method), 7  
 all() (bitbucket.issue\_comment.IssueComment method), 8  
 all() (bitbucket.repository.Repository method), 8  
 all() (bitbucket.service.Service method), 9  
 all() (bitbucket.ssh.SSH method), 9  
 AnonymousBitbucketTest (class in bitbucket.tests.public), 10  
 archive() (bitbucket.repository.Repository method), 8  
 ArchiveRepositoryAuthenticatedMethodsTest (class in bitbucket.tests.private.repository), 12  
 auth (bitbucket.bitbucket.Bitbucket attribute), 6  
 AuthenticatedBitbucketTest (class in bitbucket.tests.private.private), 11  
 authorize() (bitbucket.bitbucket.Bitbucket method), 6

**B**

Bitbucket (class in bitbucket.bitbucket), 6  
 bitbucket (module), 5  
 bitbucket.bitbucket (module), 6  
 bitbucket.issue (module), 7  
 bitbucket.issue\_comment (module), 8  
 bitbucket.repository (module), 8  
 bitbucket.service (module), 9  
 bitbucket.ssh (module), 9  
 bitbucket.tests (module), 10  
 bitbucket.tests.private (module), 11  
 bitbucket.tests.private.issue (module), 11  
 bitbucket.tests.private.issue\_comment (module), 11  
 bitbucket.tests.private.private (module), 11  
 bitbucket.tests.private.repository (module), 12  
 bitbucket.tests.private.service (module), 12  
 bitbucket.tests.private.ssh (module), 13  
 bitbucket.tests.public (module), 10  
 BitbucketAnonymousMethodsTest (class in bitbucket.tests.public), 10  
 BitbucketAuthenticatedMethodsTest (class in bitbucket.tests.private.private), 11

BitbucketUtilitiesTest (class in bitbucket.tests.public), 10

**C**

create() (bitbucket.issue.Issue method), 7  
 create() (bitbucket.issue\_comment.IssueComment method), 8  
 create() (bitbucket.repository.Repository method), 8  
 create() (bitbucket.service.Service method), 9  
 create() (bitbucket.ssh.SSH method), 9

**D**

delete() (bitbucket.issue.Issue method), 7  
 delete() (bitbucket.issue\_comment.IssueComment method), 8  
 delete() (bitbucket.repository.Repository method), 9  
 delete() (bitbucket.service.Service method), 9  
 delete() (bitbucket.ssh.SSH method), 9  
 dispatch() (bitbucket.bitbucket.Bitbucket method), 6

**F**

finalize\_oauth() (bitbucket.bitbucket.Bitbucket method), 6

**G**

get() (bitbucket.issue.Issue method), 7  
 get() (bitbucket.issue\_comment.IssueComment method), 8  
 get() (bitbucket.repository.Repository method), 9  
 get() (bitbucket.service.Service method), 9  
 get() (bitbucket.ssh.SSH method), 9  
 get\_branches() (bitbucket.bitbucket.Bitbucket method), 7  
 get\_privileges() (bitbucket.bitbucket.Bitbucket method), 7  
 get\_tags() (bitbucket.bitbucket.Bitbucket method), 7  
 get\_user() (bitbucket.bitbucket.Bitbucket method), 7

**I**

Issue (class in bitbucket.issue), 7  
 issue\_id (bitbucket.issue.Issue attribute), 7

IssueAuthenticatedMethodsTest (class in bitbucket.tests.private.issue), 11  
 IssueComment (class in bitbucket.issue\_comment), 8  
 IssueCommentAuthenticatedMethodsTest (class in bitbucket.tests.private.issue\_comment), 11

## P

password (bitbucket.bitbucket.Bitbucket attribute), 7  
 public() (bitbucket.repository.Repository method), 9

## R

repo\_slug (bitbucket.bitbucket.Bitbucket attribute), 7  
 Repository (class in bitbucket.repository), 8  
 RepositoryAuthenticatedMethodsTest (class in bitbucket.tests.private.repository), 12

## S

Service (class in bitbucket.service), 9  
 ServiceAuthenticatedMethodsTest (class in bitbucket.tests.private.service), 12  
 setUp() (bitbucket.tests.private.issue\_comment.IssueCommentAuthenticatedMethodsTest method), 11  
 setUp() (bitbucket.tests.private.private.AuthenticatedBitbucketTest method), 11  
 setUp() (bitbucket.tests.private.repository.ArchiveRepositoryAuthenticatedMethodsTest method), 12  
 setUp() (bitbucket.tests.public.AnonymousBitbucketTest method), 10  
 skipUnlessHasGit() (in module bitbucket.tests.private.repository), 12  
 SSH (class in bitbucket.ssh), 9  
 SSHAuthenticatedMethodsTest (class in bitbucket.tests.private.ssh), 13

## T

tearDown() (bitbucket.tests.private.issue\_comment.IssueCommentAuthenticatedMethodsTest method), 11  
 tearDown() (bitbucket.tests.private.private.AuthenticatedBitbucketTest method), 11  
 tearDown() (bitbucket.tests.private.repository.ArchiveRepositoryAuthenticatedMethodsTest method), 12  
 tearDown() (bitbucket.tests.public.AnonymousBitbucketTest method), 10  
 test\_all() (bitbucket.tests.private.issue.IssueAuthenticatedMethodsTest method), 11  
 test\_all() (bitbucket.tests.private.issue\_comment.IssueCommentAuthenticatedMethodsTest method), 11  
 test\_all() (bitbucket.tests.private.repository.RepositoryAuthenticatedMethodsTest method), 12  
 test\_all() (bitbucket.tests.private.service.ServiceAuthenticatedMethodsTest method), 12  
 test\_all() (bitbucket.tests.private.ssh.SSHAuthenticatedMethodsTest method), 13

test\_archive() (bitbucket.tests.private.repository.ArchiveRepositoryAuthenticatedMethodsTest method), 12  
 test\_auth() (bitbucket.tests.public.BitbucketUtilitiesTest method), 10  
 test\_create() (bitbucket.tests.private.repository.RepositoryAuthenticatedMethodsTest method), 12  
 test\_CRUD() (bitbucket.tests.private.issue.IssueAuthenticatedMethodsTest method), 11  
 test\_CRUD() (bitbucket.tests.private.issue\_comment.IssueCommentAuthenticatedMethodsTest method), 11  
 test\_CRUD() (bitbucket.tests.private.service.ServiceAuthenticatedMethodsTest method), 12  
 test\_CRUD() (bitbucket.tests.private.ssh.SSHAuthenticatedMethodsTest method), 13  
 test\_default\_credential() (bitbucket.tests.public.BitbucketUtilitiesTest method), 10  
 test\_delete() (bitbucket.tests.private.repository.RepositoryAuthenticatedMethodsTest method), 12  
 test\_dispatch\_delete() (bitbucket.tests.public.BitbucketUtilitiesTest method), 10  
 test\_dispatch\_get() (bitbucket.tests.public.BitbucketUtilitiesTest method), 10  
 test\_dispatch\_post() (bitbucket.tests.public.BitbucketUtilitiesTest method), 10  
 test\_dispatch\_put() (bitbucket.tests.public.BitbucketUtilitiesTest method), 10  
 test\_get() (bitbucket.tests.private.repository.RepositoryAuthenticatedMethodsTest method), 12  
 test\_get\_branches() (bitbucket.tests.private.private.BitbucketAuthenticatedMethodsTest method), 12  
 test\_get\_none\_public\_repos() (bitbucket.tests.public.BitbucketAnonymousMethodsTest method), 10  
 test\_get\_none\_user() (bitbucket.tests.public.BitbucketAnonymousMethodsTest method), 10  
 test\_get\_public\_repos() (bitbucket.tests.public.BitbucketAnonymousMethodsTest method), 10  
 test\_get\_self\_public\_repos() (bitbucket.tests.public.BitbucketAnonymousMethodsTest method), 10  
 test\_get\_self\_user() (bitbucket.tests.public.BitbucketAnonymousMethodsTest method), 10  
 test\_get\_tags() (bitbucket.tests.private.private.BitbucketAuthenticatedMethodsTest method), 12  
 test\_get\_user() (bitbucket.tests.public.BitbucketAnonymousMethodsTest method), 10



method), 10  
test\_password() (bitbucket.tests.public.BitbucketUtilitiesTest  
method), 10  
test\_repo\_slug() (bitbucket.tests.public.BitbucketUtilitiesTest  
method), 10  
test\_update() (bitbucket.tests.private.repository.RepositoryAuthenticatedMethodsTest  
method), 12  
test\_url\_complex() (bit-  
bucket.tests.public.BitbucketUtilitiesTest  
method), 10  
test\_url\_simple() (bitbucket.tests.public.BitbucketUtilitiesTest  
method), 10  
test\_username() (bitbucket.tests.public.BitbucketUtilitiesTest  
method), 11

## U

update() (bitbucket.issue.Issue method), 8  
update() (bitbucket.issue\_comment.IssueComment  
method), 8  
update() (bitbucket.repository.Repository method), 9  
update() (bitbucket.service.Service method), 9  
url() (bitbucket.bitbucket.Bitbucket method), 7  
username (bitbucket.bitbucket.Bitbucket attribute), 7

## V

verify() (bitbucket.bitbucket.Bitbucket method), 7