# **Python-Bitbucket Documentation**

Release 0.5

**Baptiste Millou** 

## Contents

1	Insta	ıllation	1
	1.1	Pip	1
	1.2	Get the Code & contribute	
	1.3	Test	1
2	Usag	ge	3
	2.1	Public	3
	2.2	Private	3
	2.3	Examples	3
3	bitbu	icket Package	5
	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
4	Indic	ces and tables	15
Ру	thon I	Module Index	17

## CHAPTER 1

Installation

## **Pip**

Installing Python Bitbucket is simple with pip:

pip install git+git@github.com:Alir3z4/python-bitbucket.git#egg=bitbucket

## **Get the Code & contribute**

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

You can either clone the public repository:

git clone git@github.com:Alir3z4/python-bitbucket.git

#### Download the tarball:

curl -OL https://github.com/Alir3z4/python-bitbucket/tarball/master

### Or, download the zipball:

curl -OL https://github.com/Alir3z4/python-bitbucket/zipball/master

## **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

## CHAPTER 2

Usage

## **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=USERNAME, repo_name_or_slug='public_slug')
>>> success, result = bb.repository.delete()
>>> print success
False
```

## **Private**

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

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

## **Examples**

Connect using Oauth

```
>>> import webbrowser
>>> from bitbucket.bitbucket import Bitbucket
>>> bb = Bitbucket(username=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=USERNAME, password=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)))
```

4 Chapter 2. Usage

## CHAPTER 3

bitbucket Package

## 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)
```

### 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.

 ${\tt finalize\_oauth}~(access\_token, access\_token\_secret)$ 

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 privledges 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.

```
url_v2 (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.

### 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, owner=None)
```

Get issues from one of your repositories.

```
create (repo_slug=None, owner=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, owner=None)

Delete an issue from one of your repositories.

3.3. issue Module 7

```
get (issue_id, repo_slug=None, owner=None)
```

Get an issue from one of your repositories.

#### issue\_id

Your repository slug name.

```
update (issue_id, repo_slug=None, owner=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).

## 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, owner=None)

Get issue comments from one of your repositories.

```
create (issue_id=None, repo_slug=None, owner=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, owner=None)

Delete an issue from one of your repositories.

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

Get an issue from one of your repositories.

```
update (comment id, issue id=None, repo slug=None, owner=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.

## repository Module

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

This class provide repository-related methods to Bitbucket objects.

all(owner=None)

Return all repositories owned by a given owner

```
archive (repo_slug=None, owner=None, format='zip', prefix='')
```

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

format parameter is curently not supported.

```
create (repo_name=None, repo_slug=None, owner=None, scm='git', private=True, **kwargs)
Creates a new repository on a Bitbucket account and return it.
```

#### delete (repo\_slug=None, owner=None)

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

```
fork (accountname, repo slug, name, **kwargs)
```

Fork repository on {Bitbucket accountname}/repo\_slug to {own Bitbucket account}/name and return it.

```
get (repo_slug=None, owner=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.

#### team (include\_owned=True)

Return all repositories for which the authenticated user is part of the team.

If include\_owned is True (default), repos owned by the user are included (and therefore is a superset of the repos returned by all().

If include\_owned is False, repositories only repositories owned by other users are returned.

```
update (repo_slug=None, owner=None, **kwargs)
```

Updates repository on a Bitbucket account and return it.

### 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 confimation 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.

### 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.

3.6. service Module 9

```
create (kev=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 confimation
         and NO undo.
     get (key id=None)
          Get one of the ssh keys associated with your account.
Subpackages
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 annonymous Bitbucket...
     tearDown()
         Destroy the Bitbucket...
class bitbucket.tests.public.BitbucketAnnonymousMethodsTest (methodName='runTest')
     Bases: bitbucket.tests.public.AnonymousBitbucketTest
     Test Bitbucket annonymous 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
{\bf class} \ {\tt bitbucket.tests.private.issue\_comment.IssueCommentAuthenticatedMethodsTest} \ (\textit{methodName} = `rname{'rname}' \ (\textit{methodName} = `rname{''name}' \ (\textit{methodName} = `rname{''
                 \textbf{Bases:} \ \textit{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()
```

3.8. Subpackages 11

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 annonymous methods.
     test_get_branches()
         Test get_branches.
     test_get_tags()
         Test get_tags.
repository Module
class bitbucket.tests.private.repository.ArchiveRepositoryAuthenticatedMethodsTest (methodName=
     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_fork()
```

Test repository fork.

Test get a repository.

test\_get()

```
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.
```

### settings Module

test\_all()

Test get all services.

Set **USERNAME** and **PASSWORD** or **USERNAME**, **CONSUMER\_KEY** and **CONSUMER\_SECRET** in *bit-bucket/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.
```

3.8. Subpackages 13

# $\mathsf{CHAPTER}\, 4$

## Indices and tables

- genindex
- modindex
- search

## Python Module Index

## 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,
bitbucket.tests.private.private, 12
bitbucket.tests.private.repository, 12
bitbucket.tests.private.service, 13
bitbucket.tests.private.ssh, 13
bitbucket.tests.public, 10
```

18 Python Module Index

## Index

A	BitbucketUtilitiesTest (class in bitbucket.tests.public), 10	
all() (bitbucket.issue.Issue method), 7 all() (bitbucket.issue_comment.IssueComment method), 8	C create() (bitbucket.issue.Issue method), 7	
all() (bitbucket.repository.Repository method), 8	create() (bitbucket.issue_comment.IssueComment	
all() (bitbucket.service.Service method), 9	method), 8 create() (bitbucket.repository.Repository method), 9	
all() (bitbucket.ssh.SSH method), 9 AnonymousBitbucketTest (class in bit-	create() (bitbucket.service.Service method), 9	
bucket.tests.public), 10	create() (bitbucket.ssl-vice.ssl-vice method), 9	
archive() (bitbucket.repository.Repository method), 8	create() (ottoucket.ssii.ssii inealou), y	
ArchiveRepositoryAuthenticatedMethodsTest (class in	D	
bitbucket.tests.private.repository), 12	delete() (bitbucket.issue.Issue method), 7	
auth (bitbucket.bitbucket.Bitbucket attribute), 6	delete() (bitbucket.issue_comment.IssueComment	
AuthenticatedBitbucketTest (class in bit-	method), 8	
bucket.tests.private.private), 12	delete() (bitbucket.repository.Repository method), 9	
authorize() (bitbucket.bitbucket.Bitbucket method), 6	delete() (bitbucket.service.Service method), 9	
В	delete() (bitbucket.ssh.SSH method), 10	
	dispatch() (bitbucket.bitbucket.Bitbucket method), 6	
Bitbucket (class in bitbucket.bitbucket), 6	F	
bitbucket (module), 5	•	
bitbucket.bitbucket (module), 6	finalize_oauth() (bitbucket.bitbucket.Bitbucket method),	
bitbucket.issue (module), 7	6	
bitbucket.issue_comment (module), 8	fork() (bitbucket.repository.Repository method), 9	
bitbucket.repository (module), 8 bitbucket.service (module), 9	G	
bitbucket.ssh (module), 9		
bitbucket.tests (module), 10	get() (bitbucket.issue.Issue method), 7	
bitbucket.tests.private (module), 11	get() (bitbucket.issue_comment.IssueComment method),	
bitbucket.tests.private.issue (module), 11	8 act() (hithwelest remoditory Demoditory mothod) 0	
bitbucket.tests.private.issue_comment (module), 11	get() (bitbucket.repository.Repository method), 9 get() (bitbucket.service.Service method), 9	
bitbucket.tests.private.private (module), 12	get() (bitbucket.ssh.SSH method), 10	
bitbucket.tests.private.repository (module), 12	get_branches() (bitbucket.bitbucket.Bitbucket method), 7	
bitbucket.tests.private.service (module), 13	get_privileges() (bitbucket.bitbucket.Bitbucket method),	
bitbucket.tests.private.ssh (module), 13	7	
bitbucket.tests.public (module), 10	get_tags() (bitbucket.bitbucket.Bitbucket method), 7	
BitbucketAnnonymousMethodsTest (class in bit-	get_user() (bitbucket.bitbucket.Bitbucket method), 7	
bucket.tests.public), 10		
BitbucketAuthenticatedMethodsTest (class in bit-	1	
bucket.tests.private.private), 12	Issue (class in bitbucket.issue), 7	

issue_id (bitbucket.issue.Issue attribute), 8 IssueAuthenticatedMethodsTest (class in bit-	test_all() (bitbucket.tests.private.ssh.SSHAuthenticatedMethodsTest method), 13
bucket.tests.private.issue), 11 IssueComment (class in bitbucket.issue_comment), 8	test_archive() (bitbucket.tests.private.repository.ArchiveRepositoryAuthentimethod), 12
IssueCommentAuthenticatedMethodsTest (class in bit-	test_auth() (bitbucket.tests.public.BitbucketUtilitiesTest
bucket.tests.private.issue_comment), 11	method), 10
P	test_create() (bitbucket.tests.private.repository.RepositoryAuthenticatedMe
	method), 12 test_CRUD() (bitbucket.tests.private.issue.IssueAuthenticatedMethodsTest
password (bitbucket.bitbucket.Bitbucket attribute), 7 public() (bitbucket.repository.Repository method), 9	method), 11
public() (bitoucket.repository.Repository method), 9	test_CRUD() (bitbucket.tests.private.issue_comment.IssueCommentAuthen
R	method), 11
repo_slug (bitbucket.bitbucket.Bitbucket attribute), 7	$test\_CRUD() \ (bitbucket. tests. private. service. Service Authenticated Methods') \ (bitbucket. tests. private. service and the service of the service of$
Repository (class in bitbucket.repository), 8	method), 13
RepositoryAuthenticatedMethodsTest (class in bit-	test_CRUD() (bitbucket.tests.private.ssh.SSHAuthenticatedMethodsTest
bucket.tests.private.repository), 12	method), 13 test_default_credential() (bit-
S	bucket.tests.public.BitbucketUtilitiesTest
	method), 10
Service (class in bitbucket.service), 9	test_delete() (bitbucket.tests.private.repository.RepositoryAuthenticatedMe
ServiceAuthenticatedMethodsTest (class in bit-	method), 12
bucket.tests.private.service), 13 setUp() (bitbucket.tests.private.issue_comment.IssueCommenthod), 11	
setUp() (bitbuckst, tests.private.private.AuthenticatedBitbu	method), 10 cketTest
setUp() (bitbucket.tests.private.repository.ArchiveRepository	ry Authenticated Methods Test
method), 12	test_dispatch_post() (bit-
setUp() (bitbucket.tests.public.AnonymousBitbucketTest	bucket.tests.public.BitbucketUtilitiesTest
method), 10 skipUnlessHasGit() (in module bit-	method), 11
bucket.tests.private.repository), 13	test_dispatch_put() (bit-
SSH (class in bitbucket.ssh), 9	bucket.tests.public.BitbucketUtilitiesTest
SSHAuthenticatedMethodsTest (class in bit-	method), 11
bucket.tests.private.ssh), 13	test_fork() (bitbucket.tests.private.repository.RepositoryAuthenticatedMeth method), 12
<b>T</b>	test_get() (bitbucket.tests.private.repository.RepositoryAuthenticatedMetho
Т	method), 12
team() (bitbucket.repository.Repository method), 9	test_get_branches() (bit-
tearDown() (bitbucket.tests.private.issue_comment.IssueC	ommentAuthenticatedMethodsTestvate.BitbucketAuthenticatedMethodsTest
method), 11	method), 12
tearDown() (bitbucket.tests.private.private.AuthenticatedB method), 12	
tearDown() (bitbucket.tests.private.repository.ArchiveRepo method), 12	<i>''</i>
tearDown() (bitbucket.tests.public.AnonymousBitbucketTe	test_get_none_user() (bit- est bucket.tests.public.BitbucketAnnonymousMethodsTest
method), 10	method), 10
test_all() (bitbucket.tests.private.issue.IssueAuthenticatedM	Methods Testublic repos() (bit-
method), 11	bucket.tests.public.BitbucketAnnonymousMethodsTest
test_all() (bitbucket.tests.private.issue_comment.IssueCom	mentAuthenticated Methods Test
method), 11	test_get_self_public_repos() (bit-
test_aii() (bitbucket.tests.private.repository.RepositoryAutl	nenticated Methods Tests.public.BitbucketAnnonymous Methods Test
method), 12 test_all() (bitbucket.tests.private.service.ServiceAuthentica	method), 10
method), 13	hucket tests public Bitbucket Annonymous Methods Test

20 Index

```
method), 10
test\_get\_tags() \ (bitbucket.tests.private.private.BitbucketAuthenticatedMethodsTest
          method), 12
test\_get\_user() \, (bitbucket.tests.public.BitbucketAnnonymousMethodsTest
          method), 10
test_password() (bitbucket.tests.public.BitbucketUtilitiesTest
          method), 11
test_repo_slug() (bitbucket.tests.public.BitbucketUtilitiesTest
          method), 11
test\_update() \ (bitbucket.tests.private.repository. Repository Authenticated Methods Test
          method), 12
test_url_complex()
                                                      (bit-
          bucket.tests.public.BitbucketUtilitiesTest\\
          method), 11
test\_url\_simple() (bitbucket.tests.public.BitbucketUtilitiesTest
          method), 11
test_username() (bitbucket.tests.public.BitbucketUtilitiesTest
          method), 11
U
update() (bitbucket.issue.Issue method), 8
               (bitbucket.issue_comment.IssueComment
update()
          method), 8
update() (bitbucket.repository.Repository method), 9
update() (bitbucket.service.Service method), 9
url() (bitbucket.bitbucket.Bitbucket method), 7
url_v2() (bitbucket.bitbucket.Bitbucket method), 7
username (bitbucket.bitbucket.Bitbucket attribute), 7
V
verify() (bitbucket.bitbucket.Bitbucket method), 7
```

Index 21