

USER GUIDE

**Testgrid API Documentation**

Arkena  
27, bld Hippolyte Marquès  
94200 Ivry sur Seine   
FRANCE

Last Update: 08.2014

Table of Contents

[Testgrid 1](#_Toc396425175)

[Client Instanciation 1](#_Toc396425176)

[Session creation 1](#_Toc396425177)

[Deploy Package 1](#_Toc396425178)

[Undeploy a specific session 2](#_Toc396425179)

[List node 2](#_Toc396425180)

[Allocation 2](#_Toc396425181)

[Release node 2](#_Toc396425182)

[Node methods 3](#_Toc396425183)

[Ansible integration 3](#_Toc396425184)

[Implementation of testgrid.model 3](#_Toc396425185)

# Testgrid

You can checkout the code at this address: git@github.com:SmartJog/Testgrid.git

## 

## Client Instanciation

user = testgrid.client.User("toto")

client = testgrid.rest.client(user = user, host = "127.0.0.1:3000")

## Session creation

session = client.open\_session("test")

## Deploy Package

Getting a package from a specific type.

Installing all the package in nodes assigned to a specific session.

package = client.get\_package("DebianPackage", pkg)

plans = session.deploy(package)

for pkg, node in plan:

print "package", pkg , "installed on node", node

## Undeploy a specific session

Uninstall and release all the nodes in a specific session.

session.undeploy()

## List node

for node in client.get\_nodes():

print node.name

## Allocation

Allocate a node according the required options.

\_opts = { “sysname” : “wheezy64” }

node = session.allocate\_node(\*\*\_opts)

## Release node

node = client.get\_node(name)

session.release(node)

## 

## Node methods

pkg = testgrid.debian.Package(name = "fleche", version = "1.0")

if node.is\_installed(pkg):

node.uninstall(pkg)

elif node.is\_installable(pkg):

node.install(pkg)

## Ansible integration

Run ansible-playbook through Testgrid

session = client.open\_session(name)

playbook = testgrid.anspkg.Playbook(pkg\_name = "motherbrain", session)

playbook.run()

## Implementation of testgrid.model

#### Grid Implementation

There are various type of grid:

* Persistent grid (grid that store all Testgrid data in sqlite3 database)
* Vagrant grid (Inherite from Persistent grid. Generate Vagrant nodes).

For example, the module testgrid.vgadapter.grid creates Vagrant nodes.

def \_create\_node(self, pkg = None, \*\*opts):

"""

Override to create a new node (hereafter called a "transient node")

\* compatible with package $pkg

\* supporting specified options $opts

Warning: do not call this directly; it's invoked by find\_node on demand.

"""

#### Node Implementation

In order to implement your own Node, you can override those abstract methods.

class Node(object):

"a node abstracts any object supporting packages & services"

\_\_metaclass\_\_ = abc.ABCMeta

def \_\_init\_\_(self, name):

self.name = name

def \_\_str\_\_(self):

return self.name

def get\_info(self):

return "no details"

@abc.abstractmethod

def has\_support(self, \*\*opts):

"return True if all specified options are supported"

pass

@abc.abstractmethod

def get\_load(self):

"return a float as a composition of load measures"

pass

@abc.abstractmethod

def join(self, subnet):

"setup a network interface on the specified subnet"

pass

@abc.abstractmethod

def leave(self, subnet):

"remove the network interface on the specified subnet"

pass

@abc.abstractmethod

def get\_subnets(self):

"return the list of subnets the node belongs to"

pass

@abc.abstractmethod

def get\_hoststring(self):

"return node hoststring as [user[:pass]@]hostname[:port]"

@abc.abstractmethod

def get\_installed\_packages(self):

" get installed package on the node"

### 

#### Package Implementation

In order to implement your own package, you can override those abstract methods.

@abc.abstractmethod

def install(self, node):

"install package on $node, raise exception on error"

pass

@abc.abstractmethod

def uninstall(self, node):

"uninstall package from $node, raise exception on error"

pass

@abc.abstractmethod

def is\_installed(self, node):

"return True if the package is installed on $node, False otherwise"

pass

@abc.abstractmethod

def is\_installable(self, node):

#### 

#### Example Implementation Package Debian

class Package(model.Package):

"debian package management commands"

def \_\_init\_\_(self, \*args, \*\*kwargs):

super(Package, self).\_\_init\_\_(\*args, \*\*kwargs)

if self.version:

self.tag = "%s=%s" % (self.name, self.version)

else:

self.tag = self.name

def install(self, node):

return self.\_run\_apt(node = node, state = "present force=yes")

def uninstall(self, node):

return self.\_run\_apt(node = node, state = "absent purge=yes force=yes")

def is\_installed(self, node):

code, stdout, stderr = node.execute("dpkg -s %s" % self.name)

return code == 0