sshed Documentation

Release 0.3.2

Colin Wood

March 23, 2015

Contents

Welcome to sshed. The simple to use paramiko wrapper. This little library makes working with ssh through python like it is working with OpenSSH on your server, laptop, or anything else that supports OpenSSH2.

Running of the base sshed server with ssh keys.

```
from sshed import servers
server = servers.from_conf('development')
serve.run('git clone git@github.com:cwood/mysite.com.git', echo=True)
>> Cloning down ...
```

Using the CentOS server with a custom run method called yum

```
from sshed.servers.centos import CentOS
server = servers.from_conf('development', server_cls=CentOS)
server.yum('install', 'python')
```

Creating a base server without the config.

```
from sshed.servers import Server

server = Server('development.mycompany.com',
                username='cwood',
                password='supersecretpassword',
                port=2222,
                compress=True)

server.run('whoami', echo=True)
>> cwood
server.run('hostname', echo=True)
>> development.mycompany.com
```

Retrying a command that has failed.

```
from sshed import servers
server = servers.from_conf('development')
command = server.run('touch /etc/httpd/extra/myhost.conf')

if command.returncode not 0:
    command.retry()
```

class sshed.servers.base.**Server**(*hostname*, *user=None*, *password=None*, ***kwargs*)

Server is a base class to call ssh commands on. It should used like this. The server object should be the base of all environment variables for a particular server. The beauty of this is that this is self contained and can be used with other tools like celery or gevent.

```
from sshed.servers import Server
development = Server('development.mycompany.com',
username='myusername',
password='mypassword')

development.run('git clone git@github.com:cwood/mysite.com.git')
development.run('sudo apachectl restart')
```

commands(*string*, *echo=False*, *raise_on_failure=True*)

Use triple quoted strings to send in a mass of shell commands. This comes in handy if you need to run a small bash script but don't want to do server.run(commanda ... b ... c) in mutiple lines.

download(*remote_path*, *local_file*)

Download a file from the remote server

file_exists(*remote_path*)
Check to see if a path exsits on the remote server

path_exists(*remote_path*)
Check to see if a path exsits on the remote server

run(*command*, *pty=False*, *echo=False*)
run should not treat sudo commands any different then normal user commands.

Need to exapnd this for failed sudo passwords and refreshing the channel.

upload(*local_file*, *remote_path*)
Upload a file to the remote server

CHAPTER 1

Indices and tables

- *genindex*
- *modindex*
- *search*

S

C

D

F

P

R

S

U