python-ssh

Release 0.1.1

Deric Degagne

Nov 22, 2021

CONTENTS

Welcome to Python-SSH's documentation.

The purpose of `Python-SSH` was to provide a convenient and easy-to-user interface for interacting with remote hosts. It makes use of the paramiko framework to establish SSH connections either directly or through tunnels to execute commands over SSH or SFTP.

CHAPTER

ONE

INSTALLATION

To get started with Python-SSH, install the latest stable release via pip:

Listing 1: Bash

```
pip install python-ssh
```

Python-SSH currently supports Python 3.6+ and relies on the following dependencies:

- paramiko
- rich

CHAPTER

TWO

GUIDE

2.1 Quickstart

This guide will walk you through the basics of how to create SSH connections, with and/or without tunnels, and open SFTP sessions.

2.1.1 Connection Methods

In order to establish a SSH connection to a remote host, user's can use one of the following 3 methods:

1. *Direct Connection*
2. *Decorated Connection*
3. *Context Connection*

See also:

For a full reference on the available configuration properties, see the *ssh2.config.SSHConfigData*.

Direct Connection

A **direct connection**, instantiates a new *ssh2.SSH*, by invoking the *ssh2.SSH* directly.

```
"""
Examples for ssh2.SSH

The following examples, create a new :class:`ssh2.SSH`, executes the command
and returns a 2-tuple with the STDOUT and exit status code.
"""

from ssh2 import SSH, SSHConfigData, SSHConnectionError


HOSTNAME = "example.com"
USERNAME = "user"
PASSWORD = "password"


def ssh_ex1():
    """
```

(continues on next page)

(continued from previous page)

```
    SSH example using :class:`ssh2.SSH` with SSH configuration
    file (:code:`~/.ssh/config`).
    """
    try:
        ssh = SSH()
        configs = SSHConfigData(hostname=HOSTNAME)
        ssh.connect(configs)
        return ssh.execute("ls -l")
    except SSHConnectionError as err:
        print(f"SSH error: {err}")


def ssh_ex2():
    """
    SSH example using :class:`ssh2.SSH` with manual SSH configuration
    arguments.
    """
    try:
        ssh = SSH()
        configs = SSHConfigData(
            hostname=HOSTNAME,
            username=USERNAME,
            password=PASSWORD,
            use_ssh_config=False
        )
        ssh.connect(configs)
        return ssh.execute("ls -l")
    except SSHConnectionError as err:
        print(f"SSH error: {err}")


if __name__ == "__main__":
    ssh_ex1()
    ssh_ex2()
```

Decorated Connection

A **decorated connection**, instantiates a new *ssh2.SSH*, by invoking the *ssh2.core.SSHConnect* object. Invocation of this object will automatically include the *ssh2.SSH* object into the decorated function.

For examples, please see SSHConnect

```
"""
Examples for ssh2.SSHConnect

The following examples, create a new :class:`ssh2.SSH`, executes the command
and returns a 2-tuple with the STDOUT and exit status code.
"""

from ssh2 import SSH, SSHConnect
```

(continues on next page)

(continued from previous page)

```
HOSTNAME = "example.com"
USERNAME = "user"
PASSWORD = "password"
CONFIGS  = {
    "hostname": HOSTNAME,
    "username": USERNAME,
    "password": PASSWORD,
    "use_ssh_config": False
}


@SSHConnect(hostname=HOSTNAME)
def sshconnect_ex1(ssh: SSH):
    """
    SSH example using :class:`ssh2.SSHConnect` with SSH configuration
    file (:code:`~/.ssh/config`).
    """
    return ssh.execute("ls -l")


@SSHConnect(**CONFIGS)
def sshconnect_ex2(ssh: SSH):
    """
    SSH example using :class:`ssh2.SSHConnect` with manual configuration
    arguments.
    """
    return ssh.execute("ls -l")


if __name__ == "__main__":
    sshconnect_ex1()
    sshconnect_ex2()
```

Context Connection

A **context connection**, instantiates a new *ssh2.SSH*, by invoking the *ssh2.core.SSHContext* object.

```
"""
Examples for ssh2.SSHContext

The following examples, create a new :class:`ssh2.SSH`, executes the command
and returns a 2-tuple with the STDOUT and exit status code.
"""

from ssh2 import SSH, SSHContext


HOSTNAME = "example.com"
USERNAME = "user"
KEY_FILENAME = ["~/.ssh.id_rsa.user"]
```

(continues on next page)

(continued from previous page)

```
CONFIGS  = {
    "hostname": HOSTNAME,
    "username": USERNAME,
    "key_filename": KEY_FILENAME,
    "use_ssh_config": False
}


def sshcontext_ex1():
    """
    SSHContext example using SSH configuration file (~/.ssh/config).
    """
    with SSHContext(hostname=HOSTNAME) as ssh:
        return ssh.execute("ls -l")


def sshcontext_ex2():
    """
    SSHContext example using manual configurations.
    """
    with SSHContext(**CONFIGS) as ssh:
        return ssh.execute("ls -l")


if __name__ == "__main__":
    sshcontext_ex1()
    sshcontext_ex2()
```

Also, available within the **context connection**, is the *ssh2.core.SSHTunnelContext*. Similarly, to the *ssh2.core.SSHContext*, this instantiates a new *ssh2.SSH*, but through a tunnel connection using a jumphost. It's recommended to configure the `ProxyCommand` directive within your SSH configuration file, however, this provides the same functionality.

```
"""
Examples for ssh2.SSHTunnelContext.

The following examples, create a new :class:`ssh2.SSH` (via tunnel),
executes the command and returns a 2-tuple with the STDOUT and exit
status code.
"""

from ssh2 import SSHTunnelContext, SSHConfigData


HOSTNAME = "host1.example.com"
USERNAME = "jsmith"
KEY_FILENAME = ["~/.ssh/id_rsa.host1"]
CONFIGS  = {
    "hostname": HOSTNAME,
    "username": USERNAME,
    "key_filename": KEY_FILENAME,
    "use_ssh_config": False
```

(continues on next page)

(continued from previous page)

```
}

JUMP_HOSTNAME = "jump1.example.com"
JUMP_USERNAME = "jsmith"
JUMP_PASSWORD = "p@ssw0rd"
JUMP_CONFIGS  = {
    "hostname": JUMP_HOSTNAME,
    "username": JUMP_USERNAME,
    "password": JUMP_PASSWORD,
    "use_ssh_config": False
}


def sshtunnelcontext_ex1():
    """
    SSH Tunnel example using :class:`ssh2.SSHTunnelContext` with SSH
    configuration file (:code:`~/.ssh/config`).
    """
    with SSHTunnelContext(
        SSHConfigData(hostname=JUMP_HOSTNAME),
        SSHConfigData(hostname=HOSTNAME)) as ssh:
        return ssh.execute("ls -l")


def sshtunnelcontext_ex2():
    """
    SSH Tunnel example using :class:`ssh2.SSHTunnelContext` with manual
    configuration arguments.
    """
    with SSHTunnelContext(
        SSHConfigData(**JUMP_CONFIGS),
        SSHConfigData(**CONFIGS)) as ssh:
        return ssh.execute("ls -l")


if __name__ == "__main__":
    sshtunnelcontext_ex1()
    sshtunnelcontext_ex2()
```

2.1.2 SFTP Connection

To initiate an SFTP session, create a new *ssh2.SSH*, the open the SFTP session with *ssh2.SSH.open_sftp*.

```
"""
Examples for ssh2.SSH (SFTP)

The following examples, create a new :class:`ssh2.SSH`, opens an SFTP session
and returns the output.
"""

from ssh2 import SSH, SSHConfigData, SSHConnect, SSHContext
```

(continues on next page)

(continued from previous page)

```
HOSTNAME = "example.com"


def sftp_ex1():
    """
    SSH example using :class:`ssh2.SSH` with SSH configuration
    file (:code:`~/.ssh/config`).
    """
    ssh = SSH()
    configs = SSHConfigData(hostname=HOSTNAME)
    ssh.connect(configs)
    sftp = ssh.open_sftp()
    return sftp.listdir()


@SSHConnect(hostname=HOSTNAME)
def sftp_ex2(ssh: SSH):
    """
    SFTP example using :class:`ssh2.SSHConnect` with SSH configuration
    file (:code:`~/.ssh/config`).
    """
    sftp = ssh.open_sftp()
    return sftp.listdir()


def sftp_ex3():
    """
    SFTP example using :class:`ssh2.SSHContext` with SSH configuration
    file (:code:`~/.ssh/config`).
    """
    with SSHContext(hostname=HOSTNAME) as ssh:
        sftp = ssh.open_sftp()
        return sftp.listdir()


if __name__ == "__main__":
    sftp_ex1()
    sftp_ex2()
    sftp_ex3()
```

2.2 API Reference

2.2.1 SSH

class ssh2.**SSH**

A high-level representation of a session with an SSH server.

This class wraps the `paramiko.SSHClient` object to simplify most aspects of interacting with an SSH server.

connect(*configs:* ssh2.config.SSHConfigData) → None

Establishes SSH connection to server.

Connects to SSH server and authenticates following order of priority set by paramiko – see `paramiko.connect`.

Parameters **configs** – *ssh2.config.SSHConfigData* object.

Returns None.

disconnect() → NoReturn

Close SSH connection.

Terminates SSH connection and its underlying `paramiko.Transport`.

execute(*command: str*, *file: Union[None, TextIO] = None*) → Tuple[str, int]

Execute a command on the SSH server.

The command's output stream is returned along with the exit status code.

Parameters

- **command** – command to execute.
- **file** – an optional file-pointer object to write the output to.

Returns a 2-tuple with the STDOUT and exit status code of the executing command.

execute_realtime(*command: str*) → int

Execute a command on the SSH server.

The command's output stream is immediately printed to the terminal.

Parameters **command** – command to execute.

Returns exit status code of the executing command

open_sftp()

Open an SFTP session on the SSH server.

Note: This exposes the `paramiko.open_sftp` session object, please view the documentation at http://docs.paramiko.org/en/stable/api/sftp.html.

Returns `paramiko.SFTPClient` session object

open_tunnel(*configs:* ssh2.config.SSHConfigData, *dest_hostname: str*, *dest_port: Optional[int] = 22*) → paramiko.channel.Channel

Requests a new channel through an intermediary host.

Creates a socket-like object used for establish connects to unreachable hosts, similarily to how the `paramiko.ProxyCommand` works.

Parameters

- **dest_hostname** – The destination hostname of this port forwarding.
- **dest_port** – The destination port. Default 22.
- **configs** – *ssh2.config.SSHConfigData* object.

Returns `paramiko.Channel` object.

Raises *ssh2.errors.SSHChannelError*

2.2.2 Core Functionality

class ssh2.core.**SSHConnect**(***configs: dict*)
Creates a new `ssh2.SSH` into the decorated method.

This class supports all keys from *ssh2.config.SSHConfigData*, however, the `hostname` key must be provided.

class ssh2.core.**SSHContext**(***configs: dict*)
Creates a new `ssh2.SSH` as a context.

This class supports all keys from *ssh2.config.SSHConfigData*, however, the `hostname` key must be provided.

connect(*configs:* ssh2.config.SSHConfigData) → None
Establishes SSH connection to server.

Connects to SSH server and authenticates following order of priority set by paramiko – see `paramiko.connect`.

Parameters **configs** – *ssh2.config.SSHConfigData* object.

Returns None.

disconnect() → NoReturn
Close SSH connection.

Terminates SSH connection and its underlying `paramiko.Transport`.

execute(*command: str*, *file: Union[None, TextIO] = None*) → Tuple[str, int]
Execute a command on the SSH server.

The command's output stream is returned along with the exit status code.

Parameters

- **command** – command to execute.
- **file** – an optional file-pointer object to write the output to.

Returns a 2-tuple with the STDOUT and exit status code of the executing command.

execute_realtime(*command: str*) → int
Execute a command on the SSH server.

The command's output stream is immediately printed to the terminal.

Parameters **command** – command to execute.

Returns exit status code of the executing command

open_sftp()
Open an SFTP session on the SSH server.

Note: This exposes the `paramiko.open_sftp` session object, please view the documentation at http://docs.paramiko.org/en/stable/api/sftp.html.

Returns `paramiko.SFTPClient` session object

open_tunnel(*configs:* ssh2.config.SSHConfigData, *dest_hostname: str*, *dest_port: Optional[int] = 22*) → paramiko.channel.Channel

Requests a new channel through an intermediary host.

Creates a socket-like object used for establish connects to unreachable hosts, similarily to how the `paramiko.ProxyCommand` works.

Parameters

- **dest_hostname** – The destination hostname of this port forwarding.
- **dest_port** – The destination port. Default 22.
- **configs** – *ssh2.config.SSHConfigData* object.

Returns `paramiko.Channel` object.

Raises *ssh2.errors.SSHChannelError*

class ssh2.core.**SSHTunnelContext**(*tunnel_configs:* ssh2.config.SSHConfigData, *configs:* ssh2.config.SSHConfigData)

Creates a new `ssh2.SSH`` into the decorated method.

This class supports all keys from *ssh2.config.SSHConfigData*, however, the `hostname` key must be provided.

Provides an interface to create a SSH connection through a tunnel to an "unreachable" host. It's recommended that a tunnel should be completed using the SSH configuration file (ProxyCommand), however for instances where that is not possible, this context will provide this functionality.

Parameters

- **tunnel_configs** – `ssh2.SSHConfigData` object with the tunnel configurations.
- **configs** – `ssh2.SSHConfigData` object with the destination host configurations.

connect(*configs:* ssh2.config.SSHConfigData) → None

Establishes SSH connection to server.

Connects to SSH server and authenticates following order of priority set by paramiko – see `paramiko.connect`.

Parameters **configs** – *ssh2.config.SSHConfigData* object.

Returns None.

disconnect() → NoReturn

Close SSH connection.

Terminates SSH connection and its underlying `paramiko.Transport`.

execute(*command: str*, *file: Union[None, TextIO] = None*) → Tuple[str, int]

Execute a command on the SSH server.

The command's output stream is returned along with the exit status code.

Parameters

- **command** – command to execute.

- **file** – an optional file-pointer object to write the output to.

Returns a 2-tuple with the STDOUT and exit status code of the executing command.

execute_realtime(*command: str*) → int
Execute a command on the SSH server.

The command's output stream is immediately printed to the terminal.

Parameters **command** – command to execute.

Returns exit status code of the executing command

open_sftp()
Open an SFTP session on the SSH server.

Note: This exposes the `paramiko.open_sftp` session object, please view the documentation at http://docs.paramiko.org/en/stable/api/sftp.html.

Returns `paramiko.SFTPClient` session object

open_tunnel(*configs: ssh2.config.SSHConfigData*, *dest_hostname: str*, *dest_port: Optional[int] = 22*) → paramiko.channel.Channel
Requests a new channel through an intermediary host.

Creates a socket-like object used for establish connects to unreachable hosts, similarily to how the `paramiko.ProxyCommand` works.

Parameters

- **dest_hostname** – The destination hostname of this port forwarding.
- **dest_port** – The destination port. Default 22.
- **configs** – *ssh2.config.SSHConfigData* object.

Returns `paramiko.Channel` object.

Raises *ssh2.errors.SSHChannelError*

2.2.3 SSH Config

class ssh2.config.**SSHConfigData**(*hostname: str*, *port: int = 22*, *username: Optional[str] = None*, *password: Optional[str] = None*, *key_filename: Optional[str] = None*, *timeout: Optional[int] = None*, *allow_agent: bool = True*, *look_for_keys: bool = True*, *compress: bool = False*, *sock: Optional[Any] = None*, *gss_auth: bool = False*, *gss_kex: bool = False*, *gss_deleg_creds: bool = True*, *gss_host: Optional[str] = None*, *gss_trust_dns: bool = True*, *banner_timeout: Optional[int] = None*, *auth_timeout: Optional[int] = None*, *passphrase: Optional[str] = None*, *disabled_algorithms: Optional[str] = None*, *use_ssh_config: bool = True*, *host_key_policy: Callable = <class 'paramiko.client.RejectPolicy'>*, *host_key_file: Optional[str] = None*)

Key hostname The remote server to connect to.

Key port The remote server port to connect to, defaults to 22.

Key username The username to authenticate as.

Key password The password to authenticate with.

Key key_filename filename string (or list of filename strings) of optional private key(s) and/or certs to try for authentication.

Key timeout An optional timeout (in seconds) for the TCP connection.

Key allow_agent Enables/disables connecting to the SSH agent, defaults to True.

Key look_for_keys Enables/disables searching for discoverable private key files in `~/.ssh/`, defaults to True.

Key compress Enables/disables compressions, defaults to False.

Key sock A socket or socket-like object to use for communication to the target host.

Key gss_auth Enables/disables GSS-API authentication, defaults to False.

Key gss_kex Enables/disables GSS-API key exchange and user authentication, defaults to False.

Key gss_deleg_creds Enables/disables delegatation of GSS-API client credentials, defaults to True.

Key gss_host The targets name in the Kerberos database, defaults to None.

Key gss_trust_dns Indicates whether or not the DNS is trusted to securely canonicalize the name of the host being connected to, defaults to True.

Key banner_timeout An optional timeout (in seconds) to wait for the SSH banner to be presented.

Key auth_timeout An optional timeout (in seconds) to wait for an authentication response.

Key passphrase Used for decrypting private keys.

Key disabled_algorithms An optional dictionary passed directly to `paramiko.Transport` and its keyword argument of the same name.

Key use_ssh_config Enables/disables reading the SSH configuration file.

Key host_key_policy Indicates which SSH client or key policy to use, defaults to *paramiko.RejectPolicy*.

Key host_key_file Host key file to read, defaults to None.

`host_key_policy`
alias of `paramiko.client.RejectPolicy`

`load_ssh_config`(*config_file: Optional[str] = '~/.ssh/config'*) → dict
Loads SSH configuration properties.

Generates a dict with supported keyword/values from the ssh_config directives.

Parameters

- **hostname** (*str*) – The remote server to connect to.
- **config_file** (*str*) – The filename for the ssh_config. Default: `~/.ssh/config`

Returns A dict object with supported SSH configuration properties.

Raises ssh2.errors.SSHConfigurationError

2.2.4 Errors

exception ssh2.errors.**SFTPError**
Exception raised when an SSHClient object doesn't exist and the user attempts to create a new SFTPClient session object.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception ssh2.errors.**SSHChannelError**
Exception raised when paramiko.open_channel fails to create a socket object for our tunnel.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception ssh2.errors.**SSHConfigurationError**
Exception raised when one or more unsupported SSH configuration properties are invoked.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception ssh2.errors.**SSHConnectionError**
Exception raised when a connection to remote host fails.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception ssh2.errors.**SSHContextError**
Exception raised when the SSH context cannot be created.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

CHAPTER

THREE

RELEASES

Releases are listed at https://github.com/degagne/python-ssh/releases

CHAPTER

FOUR

LICENSE

Python-SSH is licensed under MIT license. See the LICENSE for more information.

PYTHON MODULE INDEX

S

INDEX