issho Documentation

Release 0.5.1

Michael Bilow

Jul 11, 2019

CONTENTS:

1 issho **1**
- 1.1 Installation . . . 1
- 1.2 Features . . . 1
- 1.3 Credits . . . 2

2 Installation **3**
- 2.1 Stable release . . . 3
- 2.2 From sources . . . 3

3 Setup **5**

4 Usage **7**
- 4.1 Configuration . . . 7
- 4.2 Basic Commands . . . 7
- 4.3 Convenience Functions . . . 8

5 issho **9**
- 5.1 issho package . . . 9

6 Contributing **15**
- 6.1 Types of Contributions . . . 15
- 6.2 Get Started! . . . 16
- 6.3 Pull Request Guidelines . . . 16
- 6.4 Tips . . . 17
- 6.5 Deploying . . . 17

7 Credits **19**
- 7.1 Development Lead . . . 19
- 7.2 Contributors . . . 19

8 History **21**
- 8.1 0.5.1 (2019-06-24) . . . 21
- 8.2 0.5.0 (2019-06-24) . . . 21
- 8.3 0.4.2 (2019-06-22) . . . 21
- 8.4 0.4.0 (2019-06-07) . . . 21
- 8.5 0.3.6 (2019-06-06) . . . 21
- 8.6 0.3.5 (2019-05-23) . . . 21
- 8.7 0.3.4 (2019-05-23) . . . 22
- 8.8 0.3.3 (2019-05-18) . . . 22
- 8.9 0.3.1 (2019-04-11) . . . 22
- 8.10 0.3.0 (2019-04-09) . . . 22

8.11 0.2.5 (2019-03-25) . . . 22
8.12 0.2.4 (2019-03-25) . . . 22
8.13 0.2.3 (2019-03-25) . . . 22
8.14 0.2.2 (2019-03-24) . . . 22
8.15 0.2.1 (2019-03-22) . . . 23
8.16 0.2.0 (2019-03-22) . . . 23
8.17 0.1.0 (2019-02-26) . . . 23

9 Indices and tables **25**

Python Module Index **27**

Index **29**

CHAPTER

ONE

ISSHO

`issho` and intuitive wrapper over paramiko for configuring and talking to a remote host. keyring is used to manage secrets locally.

`issho` is designed such that interacting with a single, heavily used remote machine should be *easy*, and working with more than one remote machine should be *simple*.

- Free software: MIT license
- Documentation: https://issho.readthedocs.io.

1.1 Installation

Install with `pip` or `conda`

```
pip install issho
```

```
conda install -c conda-forge issho
```

1.2 Features

- **Simple access to simple commands**
  - Port forwarding
  - Executing commands over ssh
  - Transferring files over sftp
  - Running a hive query

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

The sftp work and (future)testing framework is adapted from Jeff Hinrichs's excellent pysftp package, and some of the ssh framework is inspired by Colin Wood's sshed.

Shout out to Spencer Tipping, Neal Fultz, and Factual for helping me learn to write my own tools.

Thanks to Michael Vertuli for helping test.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

`issho` can be installed from either `pip` or `conda`

```
$ pip install issho
```

```
$ conda install -c conda-forge issho
```

2.2 From sources

The sources for issho can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/michaelbilow/issho
```

Or download the tarball:

```
$ curl  -OL https://github.com/michaelbilow/issho/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

SETUP

After installing `issho`, you will want to do some setup.

First, add the machine you want a profile for to your `.ssh/config`. For example, if you want to add a machine with the alias `dev` (the default for `issho`), you would add the following lines to your ssh config.

```
Host dev
    HostName your-host-name.com
    Port XXXXX
    User your_user
```

Once this is set up, you can set up passwords and common variables using the following command:

```
issho config dev
```

This command will drop you into an interactive prompt where you can enter passwords and configuration variables.

CHAPTER

FOUR

USAGE

4.1 Configuration

After installation, set up `issho` by following the instructions in setup.

4.2 Basic Commands

To use `issho` in a project:

```
from issho import Issho
```

The first thing to do:

```
devbox = Issho('dev')
```

This will set up a connection to the machine referred to as `dev` in your `.ssh/config`. Note that this will **only** work if Issho has already been configured.

To run a command on your devbox, you can do the following:

```
devbox.exec('echo "Hello, world!"')
'Hello, world!'
```

Note that the data is printed, not returned.

You can copy a file to or from your remote using `put` & `get`:

```
output_filename = 'test.txt'
copy_back_filename = 'get_test.txt'
with open(output_filename, 'w') as f:
    f.write('\n'.join(map(str, range(5))))
devbox.put(output_filename)
devbox.exec('cat {}'.format(output_filename))
devbox.get(output_filname, copy_back_filename)
for line in open(copy_back_filename):
    print(line.strip())
```

4.3 Convenience Functions

4.3.1 Shell Commands

Instead of using `devbox.exec(cmd, *args)`, you can write `devbox.cmd(*args)`:

```
devbox.touch('my_test.txt')
devbox.ls(' | grep my_test.txt')
devbox.rm('my_test.txt')
```

Underscores in the function name are converted to spaces:

```
devbox.seq_5()
```

4.3.2 Hadoop & HDFS

Hadoop functions can be accessed using the `.hadoop` or `.hdfs` methods. You do not need to prepend the dash to hadoop operations, though they will still work with it:

```
devbox.hdfs('ls /tmp | grep test')
devbox.hadoop('mkdir -p /tmp/test/')
```

`put` and `get` can also get from HDFS, if passed a qualified HDFS path, or if the *hadoop* option is passed.:

```
devbox.put('test.txt', '/tmp/my_folder/', hadoop=True)
devbox.get('hdfs:///tmp/myfile')
```

4.3.3 Hive

`issho` offers several convenience functions, including this for Hive:

```
devbox.hive('select * from burgers limit 10;')
devbox.hive('burger_query.sql')
```

Results from hive queries can be output locally by passing an output_filename:

```
devbox.hive('select stack(3, "hello", "cruel", "world") as val;', "hello.tsv")
```

4.3.4 Spark

`issho` can trigger a spark job using `spark-submit`; you can call it using `` `spark_submit `` or `spark`:

```
devbox.spark(application='test.jar', application_class='com.test.SparkWorkflow'...)
```

CHAPTER

FIVE

ISSHO

5.1 issho package

5.1.1 Submodules

5.1.2 issho.cli module

class issho.cli.**IsshoCLI**
Bases: object

CLI for Issho; right now only used for configuration

config(*profile*, *env=None*, *ssh_profile=''*, *ssh_config='~/.ssh/config'*, *rsa_id='~/.ssh/id_rsa'*)
Configures a single issho profile. Saves non-private variables to ~/.issho/conf.toml and passwords to the local keyring.

Parameters

- **profile** – name of the profile to configure
- **env** – Optional environment variable profile to draw from.
- **ssh_profile** – The name of the associated ssh config profile; defaults to the profile name if not supplied.
- **ssh_config** – the path to the ssh_config to be used for this profile
- **rsa_id** – the path to the id_rsa file to be used for this profile

static env(*env_name*)
Saves a set of variables to ~/.issho/envs.toml :param env_name: name of the environment to set up or update

static test_connection(*profile*, *kinit=True*)
Tests the connection to the specified :param profile: The name of the profile :param kinit: if True, will try to kinit using the stored password :return:

static update_variable(*profile*, *variable*, *value*)
Updates or add a single profile variable.

issho.cli.**main**()
Inititates the CLI using python-fire

5.1.3 issho.config module

issho.config.**read_issho_conf**(*profile*, *filename=PosixPath('/home/docs/.issho/conf.toml')*)
Writes issho variables out to a `.toml` file.

Parameters

- **profile** – The name of the profile to read
- **filename** – The output filename

Returns a dict of data stored with that profile in the configuration file

issho.config.**read_issho_env**(*profile*)
Reads issho environment variables to a dict :param profile: the name of the issho environment to draw from :return: a dict of data with that profile stored in the environment file

issho.config.**read_ssh_profile**(*ssh_config_path*, *profile*)
Helper method for getting data from .ssh/config

issho.config.**write_issho_conf**(*new_conf_dict*, *filename=PosixPath('/home/docs/.issho/conf.toml')*)
Updates the issho config file :param new_conf_dict: the new configuration to add :param filename: the location of the old configuration file

issho.config.**write_issho_env**(*new_env_dict*)
Save a new issho environment :param new_env_dict: the new set of environment paramters to add

5.1.4 issho.helpers module

issho.helpers.**able_to_connect**(*host*, *port*, *timeout=1.5*)
Returns true if it is possible to connect to the specified host and port, within the given timeout in seconds.

issho.helpers.**absolute_path**(*raw_path*)
Gets the string absolute path from a path object or string.

Parameters **raw_path** – a string or `pathlib.Path` object

issho.helpers.**add_arguments_to_cmd**(*cmd*, **args*)

issho.helpers.**clean_spark_options**(*spark_options*)

issho.helpers.**default_sftp_path**(*this_path*, *default_path*)
If `this_path` exists, return it as a path, else, return the `pathlib.Path.name` of the default path

issho.helpers.**get_pkey**(*key_path*)
Helper for getting an RSA key

issho.helpers.**get_user**()

issho.helpers.**issho_pw_name**(*pw_type*, *profile*)
Helper for standardizing password names

issho.helpers.**issho_ssh_pw_name**(*rsa_id*)
Helper for standardizing ssh password names

5.1.5 issho.issho module

Implementation for the `Issho` class, which implements a connection and some simple commands over `ssh`, using `keyring` to manage secrets locally.

class issho.issho.**Issho**(*profile='dev'*, *kinit=True*)

Bases: object

exec(*cmd*, **args*, *bg=False*, *debug=False*, *capture_output=False*)

Execute a command in bash over the SSH connection.

Note, this command does not use an interactive terminal; it instead uses a *non-interactive login* shell. This means (specifically) that your aliased commands will not work and only variables exported in your remote `.bashrc` will be available.

Parameters

- **cmd** – The bash command to be run remotely
- ***args** – Additional arguments to the command cmd
- **bg** – True = run in the background
- **debug** – True = print some debugging output
- **capture_output** – True = return stdout as a string

Returns

exec_bg(*cmd*, **args*, ***kwargs*)

Syntactic sugar for `exec(bg=True)`

get(*remotepath*, *localpath=None*, *hadoop=False*)

Gets the file at the remote path and puts it locally.

Parameters

- **remotepath** – The path on the remote from which to get.
- **localpath** – Defaults to the name of the remote path
- **hadoop** – Download from HDFS

get_output(*cmd*, **args*, ***kwargs*)

Syntactic sugar for `exec(capture_output=True)`

hadoop(*command*, **args*, ***kwargs*)

Execute the hadoop command :param command: :param args: :param kwargs: :return:

hdfs(**args*, ***kwargs*)

Syntactic sugar for hadoop

hive(*query*, *output_filename=None*, *remove_blank_top_line=True*)

Runs a hive query using the parameters set in .issho/config.toml

Parameters

- **query** – a string query, or the name of a query file name to run.
- **output_filename** – the (local) file to output the results of the hive query to. Adding this option will also keep a copy of the results in /tmp
- **remove_blank_top_line** – Hive usually has a blank top line when data is output, this parameter removes it.

kinit()

Runs kerberos init

local_forward(*remote_host*, *remote_port*, *local_host='0.0.0.0'*, *local_port=44556*)

Forwards a port from a remote through this Issho object. Useful for connecting to remote hosts that can only be accessed from inside a VPC of which your devbox is part.

put (*localpath*, *remotepath=None*, *hadoop=False*)
Puts the file at the local path to the remote.

Parameters

- **localpath** – The local path of the file to put to the remote
- **remotepath** – Defaults to the name of the local path
- **hadoop** – Upload to HDFS

spark (**args*, ***kwargs*)
Syntactic sugar for spark_submit

spark_submit (*spark_options=None*, *master=''*, *jars=''*, *files=''*, *driver_class_path=''*, *application_class=''*, *application=''*, *application_args=''*)
Submit a spark job.

Parameters

- **spark_options** – A dict of spark options
- **master** – syntactic sugar for the –master spark option
- **jars** – syntactic sugar for the –jars spark option
- **files** – syntactic sugar for the –files spark option
- **driver_class_path** – syntactic sugar for the –driver-class-path spark option
- **application_class** – syntactic sugar for the –class spark option
- **application** – the application to submit
- **application_args** – any arguments to be passed to the spark application

Returns

5.1.6 Module contents

issho - simple connections to remote machines

issho is a Python package providing a simple wrapper over paramiko, providing = operators interacting with remote machines

Main Features

Here are a few of the things that issho (should) do well:

- execute commands on a remote box
- transfer files to and from a remote easily
- set up an SSH tunnel through a remote
- run Hive & Spark jobs

TODOs

- make it easy to interact with hadoop
- make it easy to configure new services
- make it easy to add plugins to issho

CHAPTER

SIX

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/michaelbilow/issho/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

issho could always use more documentation, whether as part of the official issho docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/michaelbilow/issho/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *issho* for local development.

1. Fork the *issho* repo on GitHub, and install the pre-commit hooks.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/issho.git
$ pre-commit install
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv issho
$ cd issho/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 issho tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

3. The pull request should work for Python 3.5, 3.6, 3.7, and for PyPy. Check https://travis-ci.org/michaelbilow/issho/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_issho
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ punch --part patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Michael Bilow <michael.k.bilow@gmail.com>

7.2 Contributors

None yet. Why not be the first?

CHAPTER

EIGHT

HISTORY

8.1 0.5.1 (2019-06-24)

- Add `hadoop` operators
- Allow some simple runtime execution by overriding `__getattr__`
- Add new operators to docs

8.2 0.5.0 (2019-06-24)

- Error release

8.3 0.4.2 (2019-06-22)

- Add `spark` and `spark_submit` operator
- Upgrade to `paramiko >=2.5.0`, fixing bug with recent versions of `cryptography`

8.4 0.4.0 (2019-06-07)

- Switch from bumpversion to punch

8.5 0.3.6 (2019-06-06)

- Format code using black
- Update install to include conda-forge path

8.6 0.3.5 (2019-05-23)

- Delete blank top line from beeline by default.

8.7 0.3.4 (2019-05-23)

- Allow hive to output to a file
- Add environment variable profiles with `issho env`
- Update docs
- Allow users to re-use variables that have been set in previous configurations

8.8 0.3.3 (2019-05-18)

- Fix bug related to paramiko v2.4 not liking the Mac version of ssh keys.
- Added clear error messages to fix.

8.9 0.3.1 (2019-04-11)

- Fix bug regarding ssh vs local user identity

8.10 0.3.0 (2019-04-09)

- Add more configuration and reduce variables on the `Issho` object.
- Allow `prompt_toolkit>=1.0.10` to allow `jupyter` interoperability.
- Set up useful passwords using `issho config`

8.11 0.2.5 (2019-03-25)

- Clean up hive operator and sftp callback
- Note that `issho` is incompatible with `jupyter_console<6.0` and `ipython<7.0`

8.12 0.2.4 (2019-03-25)

- Fix bug in hive operator

8.13 0.2.3 (2019-03-25)

- Add `.readthedocs.yml`; docs build now passes.

8.14 0.2.2 (2019-03-24)

- Clean up docs, try to have a passing build

8.15 0.2.1 (2019-03-22)

- Add docstrings for all functions
- Add autodocs
- Switch out `bumpversion` for `bump2version`

8.16 0.2.0 (2019-03-22)

- Add Hive function
- Add configuration CLI
- Fix Travis config to Python 3.5+

8.17 0.1.0 (2019-02-26)

- First release on PyPI.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

issho, 12
issho.cli, 9
issho.config, 10
issho.helpers, 10
issho.issho, 10

INDEX

A

able_to_connect() (*in module issho.helpers*), 10
absolute_path() (*in module issho.helpers*), 10
add_arguments_to_cmd() (*in module issho.helpers*), 10

C

clean_spark_options() (*in module issho.helpers*), 10
config() (*issho.cli.IsshoCLI method*), 9

D

default_sftp_path() (*in module issho.helpers*), 10

E

env() (*issho.cli.IsshoCLI static method*), 9
exec() (*issho.issho.Issho method*), 11
exec_bg() (*issho.issho.Issho method*), 11

G

get() (*issho.issho.Issho method*), 11
get_output() (*issho.issho.Issho method*), 11
get_pkey() (*in module issho.helpers*), 10
get_user() (*in module issho.helpers*), 10

H

hadoop() (*issho.issho.Issho method*), 11
hdfs() (*issho.issho.Issho method*), 11
hive() (*issho.issho.Issho method*), 11

I

Issho (*class in issho.issho*), 10
issho (*module*), 12
issho.cli (*module*), 9
issho.config (*module*), 10
issho.helpers (*module*), 10
issho.issho (*module*), 10
issho_pw_name() (*in module issho.helpers*), 10
issho_ssh_pw_name() (*in module issho.helpers*), 10
IsshoCLI (*class in issho.cli*), 9

K

kinit() (*issho.issho.Issho method*), 11

L

local_forward() (*issho.issho.Issho method*), 11

M

main() (*in module issho.cli*), 9

P

put() (*issho.issho.Issho method*), 11

R

read_issho_conf() (*in module issho.config*), 10
read_issho_env() (*in module issho.config*), 10
read_ssh_profile() (*in module issho.config*), 10

S

spark() (*issho.issho.Issho method*), 12
spark_submit() (*issho.issho.Issho method*), 12

T

test_connection() (*issho.cli.IsshoCLI static method*), 9

U

update_variable() (*issho.cli.IsshoCLI static method*), 9

W

write_issho_conf() (*in module issho.config*), 10
write_issho_env() (*in module issho.config*), 10