# Fabric

***Release***

**Jul 31, 2019**

# Contents

# CHAPTER 1

## What is Fabric?

Fabric is a high level Python (2.7, 3.4+) library designed to execute shell commands remotely over SSH, yielding useful Python objects in return:

```
>>> from fabric import Connection
>>> result = Connection('web1.example.com').run('uname -s', hide=True)
>>> msg = "Ran {0.command!r} on {0.connection.host}, got stdout:\n{0.stdout}"
>>> print(msg.format(result))
Ran 'uname -s' on web1.example.com, got stdout:
Linux
```

It builds on top of Invoke (subprocess command execution and command-line features) and Paramiko (SSH protocol implementation), extending their APIs to complement one another and provide additional functionality.

**Note:** Fabric users may also be interested in two *strictly optional* libraries which implement best-practice user-level code: Invocations (Invoke-only, locally-focused CLI tasks) and Patchwork (remote-friendly, typically shell-command-focused, utility functions).

# CHAPTER 2

## How is it used?

Core use cases for Fabric include (but are not limited to):

- Single commands on individual hosts:

```
>>> result = Connection('web1').run('hostname')
web1
>>> result
<Result cmd='hostname' exited=0>
```

- Single commands across multiple hosts (via varying methodologies: serial, parallel, etc):

```
>>> from fabric import SerialGroup
>>> result = SerialGroup('web1', 'web2').run('hostname')
web1
web2
>>> # Sorting for consistency...it's a dict!
>>> sorted(result.items())
[(<Connection host=web1>, <Result cmd='hostname' exited=0>), ...]
```

- Python code blocks (functions/methods) targeted at individual connections:

```
>>> def disk_free(c):
...     uname = c.run('uname -s', hide=True)
...     if 'Linux' in uname.stdout:
...         command = "df -h / | tail -n1 | awk '{print $5}'"
...         return c.run(command, hide=True).stdout.strip()
...     err = "No idea how to get disk space on {}!".format(uname)
...     raise Exit(err)
...
>>> print(disk_free(Connection('web1')))
33%
```

- Python code blocks on multiple hosts:

```
>>> # NOTE: Same code as above!
>>> def disk_free(c):
...     uname = c.run('uname -s', hide=True)
...     if 'Linux' in uname.stdout:
...         command = "df -h / | tail -n1 | awk '{print $5}'"
...         return c.run(command, hide=True).stdout.strip()
...     err = "No idea how to get disk space on {}!".format(uname)
...     raise Exit(err)
...
>>> for cxn in SerialGroup('web1', 'web2', 'db1'):
...     print("{}: {}".format(cxn, disk_free(cxn)))
<Connection host=web1>: 33%
<Connection host=web2>: 17%
<Connection host=db1>: 2%
```

In addition to these library-oriented use cases, Fabric makes it easy to integrate with Invoke's command-line task functionality, invoking via a `fab` binary stub:

- Python functions, methods or entire objects can be used as CLI-addressable tasks, e.g. `fab deploy`;
- Tasks may indicate other tasks to be run before or after they themselves execute (pre- or post-tasks);
- Tasks are parameterized via regular GNU-style arguments, e.g. `fab deploy --env=prod -d`;
- Multiple tasks may be given in a single CLI session, e.g. `fab build deploy`;
- Much more - all other Invoke functionality is supported - see its documentation for details.

# CHAPTER 3

## I'm a user of Fabric 1, how do I upgrade?

We've packaged modern Fabric in a manner that allows installation alongside Fabric 1, so you can upgrade at whatever pace your use case requires. There are multiple possible approaches – see our *detailed upgrade documentation* for details.

# CHAPTER 4

## What is this website?

`www.fabfile.org` provides project information for Fabric such as the changelog, contribution guidelines, development roadmap, news/blog, and so forth.

Detailed conceptual and API documentation can be found at our code documentation site, docs.fabfile.org.

## 4.1 Changelog

**Note:** Looking for the Fabric 1.x changelog? See *Changelog (1.x)*.

- #1709: Add `Group.close` to allow closing an entire group's worth of connections at once. Patch via Johannes Löthberg.
- #1780: Add context manager behavior to `Group`, to match the same feature in `Connection`. Feature request by István Sárándi.
- #1849: Add `Connection.from_v1` (and `Config.from_v1`) for easy creation of modern `Connection`/`Config` objects from the currently configured Fabric 1.x environment. Should make upgrading piecemeal much easier for many use cases.
- #1852: Grant internal `Connection` objects created during `ProxyJump` based gateways/proxies a copy of the outer `Connection`'s configuration object. This was not previously done, which among other things meant one could not fully disable SSH config file loading (as the internal `Connection` objects would revert to the default behavior). Thanks to Chris Adams for the report.
- : Some debug logging was reusing Invoke's logger object, generating log messages "named" after `invoke` instead of `fabric`. This has been fixed by using Fabric's own logger everywhere instead.
- #1850: Skip over `ProxyJump` configuration directives in SSH config data when they would cause self-referential `RecursionError` (e.g. due to wildcard-using `Host` stanzas which include the jump server itself). Reported by Chris Adams.
- : Fix a bug preventing tab completion (using the Invoke-level `--complete` flag) from completing task names correctly (behavior was to act as if there were never any tasks present, even if there was a valid fabfile nearby).

- #1852: Grant internal `Connection` objects created during `ProxyJump` based gateways/proxies a copy of the outer `Connection`'s configuration object. This was not previously done, which among other things meant one could not fully disable SSH config file loading (as the internal `Connection` objects would revert to the default behavior). Thanks to Chris Adams for the report.
- : Some debug logging was reusing Invoke's logger object, generating log messages "named" after `invoke` instead of `fabric`. This has been fixed by using Fabric's own logger everywhere instead.
- #1850: Skip over `ProxyJump` configuration directives in SSH config data when they would cause self-referential `RecursionError` (e.g. due to wildcard-using `Host` stanzas which include the jump server itself). Reported by Chris Adams.
- : Fix a bug preventing tab completion (using the Invoke-level `--complete` flag) from completing task names correctly (behavior was to act as if there were never any tasks present, even if there was a valid fabfile nearby).
- #1852: Grant internal `Connection` objects created during `ProxyJump` based gateways/proxies a copy of the outer `Connection`'s configuration object. This was not previously done, which among other things meant one could not fully disable SSH config file loading (as the internal `Connection` objects would revert to the default behavior). Thanks to Chris Adams for the report.
- : Some debug logging was reusing Invoke's logger object, generating log messages "named" after `invoke` instead of `fabric`. This has been fixed by using Fabric's own logger everywhere instead.
- #1850: Skip over `ProxyJump` configuration directives in SSH config data when they would cause self-referential `RecursionError` (e.g. due to wildcard-using `Host` stanzas which include the jump server itself). Reported by Chris Adams.
- : Fix a bug preventing tab completion (using the Invoke-level `--complete` flag) from completing task names correctly (behavior was to act as if there were never any tasks present, even if there was a valid fabfile nearby).
- #1852: Grant internal `Connection` objects created during `ProxyJump` based gateways/proxies a copy of the outer `Connection`'s configuration object. This was not previously done, which among other things meant one could not fully disable SSH config file loading (as the internal `Connection` objects would revert to the default behavior). Thanks to Chris Adams for the report.
- : Some debug logging was reusing Invoke's logger object, generating log messages "named" after `invoke` instead of `fabric`. This has been fixed by using Fabric's own logger everywhere instead.
- #1850: Skip over `ProxyJump` configuration directives in SSH config data when they would cause self-referential `RecursionError` (e.g. due to wildcard-using `Host` stanzas which include the jump server itself). Reported by Chris Adams.
- : Fix a bug preventing tab completion (using the Invoke-level `--complete` flag) from completing task names correctly (behavior was to act as if there were never any tasks present, even if there was a valid fabfile nearby).
- : Update the new functionality added for #1826 so it uses `export`; without this, nontrivial shell invocations like `command1 && command2` end up only applying the env vars to the first command.
- #1831: Grant `Group` (and subclasses) the ability to take arbitrary keyword arguments and pass them onto the internal `Connection` constructors. This allows code such as:

```
mygroup = Group('host1', 'host2', 'host3', user='admin')
```

  which was previously impossible without manually stuffing premade `Connection` objects into `Group.from_connections`.
- #1826: Add a new Boolean configuration and `Connection` parameter, `inline_ssh_env`, which (when set to `True`) changes how Fabric submits shell environment variables to remote servers; this feature helps work around commonly restrictive `AcceptEnv` settings on SSH servers. Thanks to Massimiliano Torromeo and Max Arnold for the reports.

- #1653: Clarify `Transfer` API docs surrounding remote file paths, such as the lack of tilde expansion (a buggy and ultimately unnecessary v1 feature). Thanks to `@pint12` for bringing it up.
- #1819: Moved example code from the README into the Sphinx landing page so that we could apply doctests; includes a bunch of corrections to invalid example code! Thanks to Antonio Feitosa for the initial catch & patch.
- #1762: Fix problem where lower configuration levels' setting of `connect_kwargs.key_filename` were being overwritten by the CLI `--identity` flag's value...even when that value was the empty list. CLI-given values are supposed to win, but not quite that hard. Reported by `@garu57`.
- #1749: Improve `put` behavior when uploading to directory (vs file) paths, which was documented as working but had not been fully implemented. The local path's basename (or file-like objects' `.name` attribute) is now appended to the remote path in this case. Thanks to Peter Uhnak for the report.
- : Implement `__lt__` on `Connection` so it can be sorted; this was overlooked when implementing things like `__eq__` and `__hash__`. (No, sorting doesn't usually matter much for this object type, but when you gotta, you gotta...)
- #1653: Clarify `Transfer` API docs surrounding remote file paths, such as the lack of tilde expansion (a buggy and ultimately unnecessary v1 feature). Thanks to `@pint12` for bringing it up.
- #1819: Moved example code from the README into the Sphinx landing page so that we could apply doctests; includes a bunch of corrections to invalid example code! Thanks to Antonio Feitosa for the initial catch & patch.
- #1824: The changes implementing #1772 failed to properly account for backwards compatibility with Invoke-level task objects. This has been fixed; thanks to `@ilovezfs` and others for the report.
- #1762: Fix problem where lower configuration levels' setting of `connect_kwargs.key_filename` were being overwritten by the CLI `--identity` flag's value...even when that value was the empty list. CLI-given values are supposed to win, but not quite that hard. Reported by `@garu57`.
- #1749: Improve `put` behavior when uploading to directory (vs file) paths, which was documented as working but had not been fully implemented. The local path's basename (or file-like objects' `.name` attribute) is now appended to the remote path in this case. Thanks to Peter Uhnak for the report.
- : Implement `__lt__` on `Connection` so it can be sorted; this was overlooked when implementing things like `__eq__` and `__hash__`. (No, sorting doesn't usually matter much for this object type, but when you gotta, you gotta...)
- #1653: Clarify `Transfer` API docs surrounding remote file paths, such as the lack of tilde expansion (a buggy and ultimately unnecessary v1 feature). Thanks to `@pint12` for bringing it up.
- #1819: Moved example code from the README into the Sphinx landing page so that we could apply doctests; includes a bunch of corrections to invalid example code! Thanks to Antonio Feitosa for the initial catch & patch.
- #1824: The changes implementing #1772 failed to properly account for backwards compatibility with Invoke-level task objects. This has been fixed; thanks to `@ilovezfs` and others for the report.
- #1762: Fix problem where lower configuration levels' setting of `connect_kwargs.key_filename` were being overwritten by the CLI `--identity` flag's value...even when that value was the empty list. CLI-given values are supposed to win, but not quite that hard. Reported by `@garu57`.
- #1749: Improve `put` behavior when uploading to directory (vs file) paths, which was documented as working but had not been fully implemented. The local path's basename (or file-like objects' `.name` attribute) is now appended to the remote path in this case. Thanks to Peter Uhnak for the report.
- : Implement `__lt__` on `Connection` so it can be sorted; this was overlooked when implementing things like `__eq__` and `__hash__`. (No, sorting doesn't usually matter much for this object type, but when you gotta, you gotta...)
- #1653: Clarify `Transfer` API docs surrounding remote file paths, such as the lack of tilde expansion (a buggy and ultimately unnecessary v1 feature). Thanks to `@pint12` for bringing it up.

- #1819: Moved example code from the README into the Sphinx landing page so that we could apply doctests; includes a bunch of corrections to invalid example code! Thanks to Antonio Feitosa for the initial catch & patch.
- #1824: The changes implementing #1772 failed to properly account for backwards compatibility with Invoke-level task objects. This has been fixed; thanks to `@ilovezfs` and others for the report.
- #1766: Reinstate support for use as `python -m fabric`, which (as in v1) now behaves identically to invoking `fab`. Thanks to `@RupeshPatro` for the original patchset.
- #1772: `@hosts` is back – as a `@task`/`Task` parameter of the same name. Acts much like a per-task `--hosts`, but can optionally take dicts of `fabric.connection.Connection` kwargs as well as the typical shorthand host strings.

---

**Note:** As of this change, we are now recommending the use of the new-in-this-release Fabric-level `@task`/`Task` objects instead of their Invoke counterparts, even if you're not using the `hosts` kwarg – it will help future-proof your code for similar feature-adds later, and generally be less confusing than having mixed Invoke/Fabric imports for these object types.

---

- : Updated the minimum required Invoke version to `1.1`.
- #1753: Set one of our test modules to skip user/system SSH config file loading by default, as it was too easy to forget to do so for tests aimed at related functionality. Reported by Chris Rose.
- : The `fabric.testing.fixtures.remote` pytest fixture was found to not be properly executing expectation/sanity tests on teardown; this was an oversight and has been fixed.
- : Somehow neglected to actually add `extras_require` to our `setup.py` to enable `pip install fabric[testing]` et al. This has been fixed. We hope.
- : Minor fix to `extras_require` re: having `fabric[pytest]` encompass the contents of `fabric[testing]`.
- : Our `packages=` argument to `setuptools.setup` was too specific and did not allow for subpackages... such as the newly added `fabric.testing`. Fixed now.
- : Our packaging metadata lacked a proper `MANIFEST.in` and thus some distributions were not including ancillary directories like tests and documentation. This has been fixed.
- #1753: Set one of our test modules to skip user/system SSH config file loading by default, as it was too easy to forget to do so for tests aimed at related functionality. Reported by Chris Rose.
- : The `fabric.testing.fixtures.remote` pytest fixture was found to not be properly executing expectation/sanity tests on teardown; this was an oversight and has been fixed.
- : Our `packages=` argument to `setuptools.setup` was too specific and did not allow for subpackages... such as the newly added `fabric.testing`. Fixed now.
- : Our packaging metadata lacked a proper `MANIFEST.in` and thus some distributions were not including ancillary directories like tests and documentation. This has been fixed.
- : Minor fix to `extras_require` re: having `fabric[pytest]` encompass the contents of `fabric[testing]`.
- : Somehow neglected to actually add `extras_require` to our `setup.py` to enable `pip install fabric[testing]` et al. This has been fixed. We hope.
- : Exposed our previously internal test helpers for use by downstream test suites, as the fabric.testing subpackage.

**Note:** As this code requires non-production dependencies, we've also updated our packaging metadata to publish some setuptools "extras", `fabric[testing]` (base) and `fabric[pytest]` (for pytest users).

- #1745: Wrap any imports of `invoke.vendor.*` with `try`/`except` such that downstream packages which have removed `invoke.vendor` are still able to function by using stand-alone dependencies. Patch courtesy of Othmane Madjoudj.
- #1759: Apply the `black` code formatter to the codebase and engage it on Travis-CI. Thanks to Chris Rose.
- #1761: Integration tests were never added to Travis or ported to pytest before 2.0's release; this has been addressed.

- #1745: Wrap any imports of `invoke.vendor.*` with `try`/`except` such that downstream packages which have removed `invoke.vendor` are still able to function by using stand-alone dependencies. Patch courtesy of Othmane Madjoudj.
- #1759: Apply the `black` code formatter to the codebase and engage it on Travis-CI. Thanks to Chris Rose.
- #1761: Integration tests were never added to Travis or ported to pytest before 2.0's release; this has been addressed.

- #1740: A Python 3 wheel was not uploaded during the previous release as expected; it turned out we were lacking the typical 'build universal wheels' setting in our `setup.cfg` (due to copying it from the one other project in our family of projects which explicitly cannot build universal wheels!) This has been fixed and a proper universal wheel is now built.
- : Rewrite for 2.0! See *Upgrading from 1.x*.

## 4.2 Changelog (1.x)

**Note:** This is the changelog for the legacy 1.x version of Fabric. For the current (2.0+) changelog, please see *the main changelog*.

- : Update packaging metadata so wheel archives include the `LICENSE` file.
- #983: Move a `getpass` import inside a Windows-oriented `try`/`except ImportError` so password prompting is less likely to explode on certain systems. Thanks to `@dongweiming` for the patch.
- #1227: Remove a bash/zsh-ism from `upload_template` when backing up the target file, preventing issues on simpler remote shells. Patch courtesy of Paul Chakravarti.
- #1242: (via #1243) `rsync_project`: only supply the `-p <number>` option to generated `rsync` commands when the port number differs from the default; this allows removing `--rsh` entirely most of the time, and thus enables things like using rsync's daemon mode on the remote end. Reported & patched by Arnaud Rocher.
- #1341: (via #1586) Attempt to `rm -f` the temporary file used by `put`'s sudo mode, when exceptions are encountered; previously, the internal `sudo mv` call could potentially fail and leave the file around. Thanks to Andrei Sura for the report and Uku Loskit for the fix.

- : Update packaging metadata so wheel archives include the `LICENSE` file.

- #1475: Honor `env.timeout` when opening new remote sessions (as opposed to the initial overall connection, which already honored timeout settings.) Thanks to `@EugeniuZ` for the report & `@jrmsgit` for the first draft of the patch.

**Note:** This feature only works with Paramiko 1.14.3 and above; if your Paramiko version is older, no timeout can be set, and the previous behavior will occur instead.

- #1539: Add documentation for `env.output_prefix`. Thanks `@jphalip`.
- #1416: Add explicit "Python 2 only" note to `setup.py` trove classifiers to help signal that fact to various info-gathering tools. Patch courtesy of Gavin Bisesi.
- : Backport #1462 to 1.12.x (was previously only backported to 1.13.x.)
- #1590: Replace a reference to `fab` in a test subprocess, to use the `python -m <package>` style instead; this allows `python setup.py test` to run the test suite without having Fabric already installed. Thanks to `@BenSturmfels` for catch & patch.
- #1065: Fix incorrect SSH config reference in the docs for `env.keepalive`; it corresponds to `ServerAliveInterval`, not `ClientAliveInterval`. Credit: Harry Percival.
- #1514: Compatibility with Python 2.5 was broken by using the `format()` method of a string (only in 1.11+). Report by `@pedrudehuere`.
- #1526: Disable use of PTY and shell for a background command execution within `contrib.sed`, preventing a small class of issues on some platforms/environments. Thanks to `@doflink` for the report and Pierce Lopez for the final patch.
- #1574: `upload_project` failed for folder in current directory specified without any path separator. Thanks `@aidanmelen` for the report and Pierce Lopez for the patch.
- #1294: fix text escaping for `contains` and `append` which would fail if the text contained e.g. `>`. Thanks to `@ecksun` for report & Pierce Lopez for the patch.
- #1427: (via #1428) Locate `.pyc` files when searching for fabfiles to load; previously we only used the presence of `.py` files to determine whether loading should be attempted. Credit: Ray Chen.
- #1555: Multiple simultaneous `get` and/or `put` with `use_sudo=True` and for the same remote host and path could fail unnecessarily. Thanks `@arnimarj` for the report and Pierce Lopez for the patch.
- #1542: (via #1543) Catch Paramiko-level gateway connection errors (`ChannelError`) when raising `NetworkError`; this prevents an issue where gateway related issues were being treated as authentication errors. Thanks to Charlie Stanley for catch & patch.
- #1539: Add documentation for `env.output_prefix`. Thanks `@jphalip`.
- #1416: Add explicit "Python 2 only" note to `setup.py` trove classifiers to help signal that fact to various info-gathering tools. Patch courtesy of Gavin Bisesi.
- : Backport #1462 to 1.12.x (was previously only backported to 1.13.x.)
- #1590: Replace a reference to `fab` in a test subprocess, to use the `python -m <package>` style instead; this allows `python setup.py test` to run the test suite without having Fabric already installed. Thanks to `@BenSturmfels` for catch & patch.
- #1065: Fix incorrect SSH config reference in the docs for `env.keepalive`; it corresponds to `ServerAliveInterval`, not `ClientAliveInterval`. Credit: Harry Percival.
- #1462: Make a PyCrypto-specific import and method call optional to avoid `ImportError` problems under Paramiko 2.x. Thanks to Alex Gaynor for catch & patch!
- #1514: Compatibility with Python 2.5 was broken by using the `format()` method of a string (only in 1.11+). Report by `@pedrudehuere`.

- #1526: Disable use of PTY and shell for a background command execution within `contrib.sed`, preventing a small class of issues on some platforms/environments. Thanks to `@doflink` for the report and Pierce Lopez for the final patch.
- #1574: `upload_project` failed for folder in current directory specified without any path separator. Thanks `@aidanmelen` for the report and Pierce Lopez for the patch.
- #1294: fix text escaping for `contains` and `append` which would fail if the text contained e.g. `>`. Thanks to `@ecksun` for report & Pierce Lopez for the patch.
- #1427: (via #1428) Locate `.pyc` files when searching for fabfiles to load; previously we only used the presence of `.py` files to determine whether loading should be attempted. Credit: Ray Chen.
- #1555: Multiple simultaneous `get` and/or `put` with `use_sudo=True` and for the same remote host and path could fail unnecessarily. Thanks `@arnimarj` for the report and Pierce Lopez for the patch.
- #1542: (via #1543) Catch Paramiko-level gateway connection errors (`ChannelError`) when raising `NetworkError`; this prevents an issue where gateway related issues were being treated as authentication errors. Thanks to Charlie Stanley for catch & patch.
- #1539: Add documentation for `env.output_prefix`. Thanks `@jphalip`.
- #1416: Add explicit "Python 2 only" note to `setup.py` trove classifiers to help signal that fact to various info-gathering tools. Patch courtesy of Gavin Bisesi.
- : Backport #1462 to 1.12.x (was previously only backported to 1.13.x.)
- #1590: Replace a reference to `fab` in a test subprocess, to use the `python -m <package>` style instead; this allows `python setup.py test` to run the test suite without having Fabric already installed. Thanks to `@BenSturmfels` for catch & patch.
- #1065: Fix incorrect SSH config reference in the docs for `env.keepalive`; it corresponds to `ServerAliveInterval`, not `ClientAliveInterval`. Credit: Harry Percival.
- #1462: Make a PyCrypto-specific import and method call optional to avoid `ImportError` problems under Paramiko 2.x. Thanks to Alex Gaynor for catch & patch!
- #1495: Update the internals of `fabric.contrib.files` so its members work with SSH servers running on Windows. Thanks to Hamdi Sahloul for the patch.
- #1379: (also #1464) Clean up a lot of unused imports and similar cruft (many found via `flake8 --select E4`). Thanks to Mathias Ertl for the original patches.
- #1483: (also re: #1386, #1374, #1300) Add an FAQ about quote problems in remote `csh` causing issues with Fabric's shell-wrapping and quote-escaping. Thanks to Michael Radziej for the update.
- #1461: Update setup requirements to allow Paramiko 2.x, now that it's stable and been out in the wild for some time. Paramiko 1.x still works like it always did; the only change to Paramiko 2 was the backend moving from PyCrypto to Cryptography.

> **Warning:** If you are upgrading an existing environment, the install dependencies have changed; please see Paramiko's installation docs for details: http://www.paramiko.org/installing.html

- #1458: Detect `known_hosts`-related instances of `paramiko.SSHException` and prevent them from being handled like authentication errors (which is the default behavior). This fixes issues with incorrect password prompts or prompt-related exceptions when using `reject_unknown_hosts` and encountering missing or bad `known_hosts` entries. Thanks to Lukáš Doktor for catch & patch.
- #1470: When using `fabric.operations.get` with glob expressions, a lack of matches for the glob would result in an empty file named after the glob expression (in addition to raising an error). This has been fixed so the empty file is no longer generated. Thanks to Georgy Kibardin for the catch & initial patch.

- #1379: (also #1464) Clean up a lot of unused imports and similar cruft (many found via `flake8 --select E4`). Thanks to Mathias Ertl for the original patches.
- #1483: (also re: #1386, #1374, #1300) Add an FAQ about quote problems in remote `csh` causing issues with Fabric's shell-wrapping and quote-escaping. Thanks to Michael Radziej for the update.
- #1458: Detect `known_hosts`-related instances of `paramiko.SSHException` and prevent them from being handled like authentication errors (which is the default behavior). This fixes issues with incorrect password prompts or prompt-related exceptions when using `reject_unknown_hosts` and encountering missing or bad `known_hosts` entries. Thanks to Lukáš Doktor for catch & patch.
- #1470: When using `fabric.operations.get` with glob expressions, a lack of matches for the glob would result in an empty file named after the glob expression (in addition to raising an error). This has been fixed so the empty file is no longer generated. Thanks to Georgy Kibardin for the catch & initial patch.
- #1379: (also #1464) Clean up a lot of unused imports and similar cruft (many found via `flake8 --select E4`). Thanks to Mathias Ertl for the original patches.
- #1483: (also re: #1386, #1374, #1300) Add an FAQ about quote problems in remote `csh` causing issues with Fabric's shell-wrapping and quote-escaping. Thanks to Michael Radziej for the update.
- #1458: Detect `known_hosts`-related instances of `paramiko.SSHException` and prevent them from being handled like authentication errors (which is the default behavior). This fixes issues with incorrect password prompts or prompt-related exceptions when using `reject_unknown_hosts` and encountering missing or bad `known_hosts` entries. Thanks to Lukáš Doktor for catch & patch.
- #1470: When using `fabric.operations.get` with glob expressions, a lack of matches for the glob would result in an empty file named after the glob expression (in addition to raising an error). This has been fixed so the empty file is no longer generated. Thanks to Georgy Kibardin for the catch & initial patch.
- #1379: (also #1464) Clean up a lot of unused imports and similar cruft (many found via `flake8 --select E4`). Thanks to Mathias Ertl for the original patches.
- #1483: (also re: #1386, #1374, #1300) Add an FAQ about quote problems in remote `csh` causing issues with Fabric's shell-wrapping and quote-escaping. Thanks to Michael Radziej for the update.
- #1491: Implement `sudo`-specific password caching. This can be used to work around issues where over-eager submission of `env.password` at login time causes authentication problems (e.g. during two-factor auth).
- #1447: Fix a relative import in `fabric.network` to be correctly/consistently absolute instead. Thanks to `@bildzeitung` for catch & patch.
- #1447: Fix a relative import in `fabric.network` to be correctly/consistently absolute instead. Thanks to `@bildzeitung` for catch & patch.
- : Bumped version to `1.11.1` due to apparently accidentally uploading a false `1.11.0` to PyPI sometime in the past (PyPI is secure & prevents reusing deleted filenames.) We have no memory of this, but databases don't lie!
- #1200: Introduced `exceptions` output level, so users don't have to deal with the debug output just to see tracebacks.
- #1388: Expose Jinja's `keep_trailing_newline` parameter in `fabric.contrib.files.upload_template` so users can force template renders to preserve trailing newlines. Thanks to Chen Lei for the patch.
- #1326: Make `fabric.contrib.project.rsync_project` aware of `env.gateway`, using a `ProxyCommand` under the hood. Credit: David Rasch.
- #1271: Allow users whose fabfiles use `fabric.colors` to disable colorization at runtime by specifying `FABRIC_DISABLE_COLORS=1` (or any other non-empty value). Credit: Eric Berg.

- #1261: Expose Paramiko's Kerberos functionality as Fabric config vars & command-line options. Thanks to Ramanan Sivaranjan for catch & patch, and to Johannes Löthberg & Michael Bennett for additional testing.
- #932: Add a `temp_dir` kwarg to `fabric.contrib.files.upload_template` which is passed into its inner `fabric.operations.put` call. Thanks to `@nburlett` for the patch.
- #1161: Add `use_sudo` kwarg to `fabric.operations.reboot`. Credit: Bryce Verdier.
- #800: Add `capture_buffer_size` kwarg to `fabric.operations.run`/`fabric.operations.sudo` so users can limit memory usage in situations where subprocesses generate very large amounts of stdout/err. Thanks to Jordan Starcher for the report & Omri Bahumi for an early version of the patchset.
- #1203: (via #1240) Add a `case_sensitive` kwarg to `fabric.contrib.files.contains` (which toggles use of `egrep -i`). Report by `@xoul`, patch by Curtis Mattoon.
- #1389: Gently overhaul SSH port derivation so it's less surprising; previously, any non-default value stored in `env.port` was overriding all SSH-config derived values. See the API docs for `fabric.network.normalize` for details on how it now behaves. Thanks to Harry Weppner for catch & patch.
- #1229: Add some missing API doc hyperlink references. Thanks to Tony Narlock.
- #958: Remove the Git SHA portion of our version string generation; it was rarely useful & occasionally caused issues for users with non-Git-based source checkouts.
- #1213: Add useful exception message to the implicit `SystemExit` raised by Fabric's use of `sys.exit` inside the `fabric.api.abort` function. This allows client code catching `SystemExit` to have better introspection into the error. Thanks to Ioannis Panousis.
- #1239: Update README to work better under raw docutils so the example code block is highlighted as Python on PyPI (and not just on our Sphinx-driven website). Thanks to Marc Abramowitz.
- #1325: Clarify `fabric.operations.put` docs re: the `mode` argument. Thanks to `@mjmare` for the catch.
- #1454: Remove use of `:option:` directives in the changelog, it's currently broken in modern Sphinx & doesn't seem to have actually functioned on Renaissance-era Sphinx either.
- #1359: Add a more-visible top-level `CHANGELOG.rst` pointing users to the actual changelog stored within the Sphinx directory tree. Thanks to Jonathan Vanasco for catch & patch.
- #1257: Add notes to the usage docs for `fab` regarding the program's exit status. Credit: `@koalaman`.
- #943: Tweak `env.warn_only` docs to note that it applies to all operations, not just `run`/`sudo`. Thanks `@akitada`.
- #1348: (via #1361) Fix a bug in `fabric.operations.get` where remote file paths containing Python string formatting escape codes caused an exception. Thanks to `@natecode` for the report and Bradley Spink for the fix.
- #1365: (via #1372) Classic-style fabfiles (ones not using `@task`) erroneously included custom exception subclasses when collecting tasks. This is now fixed thanks to `@mattvonrocketstein`.
- #1135: (via #1241) Modified order of operations in `fabric.operations.run`/`fabric.operations.sudo` to apply environment vars before prefixing commands (instead of after). Report by `@warsamebashir`, patch by Curtis Mattoon.
- #1454: Remove use of `:option:` directives in the changelog, it's currently broken in modern Sphinx & doesn't seem to have actually functioned on Renaissance-era Sphinx either.
- #1257: Add notes to the usage docs for `fab` regarding the program's exit status. Credit: `@koalaman`.
- #943: Tweak `env.warn_only` docs to note that it applies to all operations, not just `run`/`sudo`. Thanks `@akitada`.

- #1273: Fix issue with ssh/config not having a cross-platform default path. Thanks to `@SamuelMarks` for catch & patch.
- #1286: (also #971, #1032) Recursively unwrap decorators instead of only unwrapping a single decorator level, when obtaining task docstrings. Thanks to Avishai Ish-Shalom for the original report & Max Kovgan for the patch.
- #1289: Fix "NameError: free variable referenced before assignment in enclosing scope". Thanks to `@SamuelMarks` for catch & patch.
- #980: (also #1312) Redirect output of `cd` to `/dev/null` so users enabling bash's `CDPATH` (or similar features in other shells) don't have polluted output captures. Thanks to Alex North-Keys for the original report & Steve Ivy for the fix.
- #1305: (also #1313) Fix a couple minor issues with the operation of & demo code for the `JobQueue` class. Thanks to `@dioh` and Horst Gutmann for the report & Cameron Lane for the patch.
- #1318: Update functionality added in #1213 so abort error messages don't get printed twice (once by us, once by `sys.exit`) but the annotated exception error message is retained. Thanks to Felix Almeida for the report.
- #1226: Update `fabric.operations.get` to ensure that `env.user` has access to tempfiles before changing permissions. Also corrected permissions from 404 to 0400 to match comment. Patch by Curtis Mattoon; original report from Daniel Watkins.
- #1180: Fix issue with unicode steam outputs crashing if stream encoding type is None. Thanks to `@joekiller` for catch & patch.
- #1228: Update the `CommandTimeout` class so it has a useful `str` instead of appearing blank when caught by Fabric's top level exception handling. Catch & patch from Tomaz Muraus.
- #1019: (also #1022, #1186) Fix "is a tty" tests in environments where streams (eg `sys.stdout`) have been replaced with objects lacking a `.isatty()` method. Thanks to Miki Tebeka for the original report, Lele Long for a subsequent patch, and Julien Phalip for the final/merged patch.
- #1201: Don't naively glob all `fabric.operations.get` targets - only glob actual directories. This avoids incorrectly yielding permission errors in edge cases where a requested file is within a directory lacking the read permission bit. Thanks to Sassa Nf for the original report.
- #1229: Add some missing API doc hyperlink references. Thanks to Tony Narlock.
- #958: Remove the Git SHA portion of our version string generation; it was rarely useful & occasionally caused issues for users with non-Git-based source checkouts.
- #1213: Add useful exception message to the implicit `SystemExit` raised by Fabric's use of `sys.exit` inside the `fabric.api.abort` function. This allows client code catching `SystemExit` to have better introspection into the error. Thanks to Ioannis Panousis.
- #1226: Update `fabric.operations.get` to ensure that `env.user` has access to tempfiles before changing permissions. Also corrected permissions from 404 to 0400 to match comment. Patch by Curtis Mattoon; original report from Daniel Watkins.
- #1180: Fix issue with unicode steam outputs crashing if stream encoding type is None. Thanks to `@joekiller` for catch & patch.
- #1228: Update the `CommandTimeout` class so it has a useful `str` instead of appearing blank when caught by Fabric's top level exception handling. Catch & patch from Tomaz Muraus.
- #1019: (also #1022, #1186) Fix "is a tty" tests in environments where streams (eg `sys.stdout`) have been replaced with objects lacking a `.isatty()` method. Thanks to Miki Tebeka for the original report, Lele Long for a subsequent patch, and Julien Phalip for the final/merged patch.

- #1201: Don't naively glob all `fabric.operations.get` targets - only glob actual directories. This avoids incorrectly yielding permission errors in edge cases where a requested file is within a directory lacking the read permission bit. Thanks to Sassa Nf for the original report.
- #1229: Add some missing API doc hyperlink references. Thanks to Tony Narlock.
- #958: Remove the Git SHA portion of our version string generation; it was rarely useful & occasionally caused issues for users with non-Git-based source checkouts.
- #1213: Add useful exception message to the implicit `SystemExit` raised by Fabric's use of `sys.exit` inside the `fabric.api.abort` function. This allows client code catching `SystemExit` to have better introspection into the error. Thanks to Ioannis Panousis.
- #975: Fabric can now be invoked via `python -m fabric` in addition to the typical use of the `fab` entrypoint. Patch courtesy of Jason Coombs.

---

**Note:** This functionality is only available under Python 2.7.

---

- #1090: Add option to skip unknown tasks. Credit goes to Jonas Lundberg.
- #1098: Add support for dict style roledefs. Thanks to Jonas Lundberg.
- #700: Added `use_sudo` and `temp_dir` params to `fabric.operations.get`. This allows downloading files normally not accessible to the user using `sudo`. Thanks to Jason Coombs for initial report and to Alex Plugaru for the patch (#1121).
- #1188: Update `fabric.operations.local` to close non-pipe file descriptors in the child process so subsequent calls to `fabric.operations.local` aren't blocked on e.g. already-connected network sockets. Thanks to Tolbkni Kao for catch & patch.
- #1167: Add Jinja to `test_requires` in `setup.py` for the couple of newish tests that now require it. Thanks to Kubilay Kocak for the catch.
- #600: Clear out connection caches in full when prepping parallel-execution subprocesses. This avoids corner cases causing hangs/freezes due to client/socket reuse. Thanks to Ruslan Lutsenko for the initial report and Romain Chossart for the suggested fix.
- #1026: Fix a typo preventing quiet operation of `fabric.contrib.files.is_link`. Caught by `@dongweiming`.
- #1059: Update IPv6 support to work with link-local address formats. Fix courtesy of `@obormot`.
- #1096: Encode Unicode text appropriately for its target stream object to avoid issues on non-ASCII systems. Thanks to Toru Uetani for the original patch.
- #852: Fix to respect `template_dir` for non Jinja2 templates in `fabric.contrib.files.upload_template`. Thanks to Adam Kowalski for the patch and Alex Plugaru for the initial test case.
- #1134: Skip bad hosts when the tasks are executed in parallel. Thanks to Igor Maravić `@i-maravic`.
- #1146: Fix a bug where `fabric.contrib.files.upload_template` failed to honor `lcd` when `mirror_local_mode` is `True`. Thanks to Laszlo Marai for catch & patch.
- #1147: Use `stat` instead of `lstat` when testing directory-ness in the SFTP module. This allows recursive downloads to avoid recursing into symlinks unexpectedly. Thanks to Igor Kalnitsky for the patch.
- #1165: Prevent infinite loop condition when a gateway host is enabled & the same host is in the regular target host list. Thanks to `@CzBiX` for catch & patch.
- #1167: Add Jinja to `test_requires` in `setup.py` for the couple of newish tests that now require it. Thanks to Kubilay Kocak for the catch.

- #600: Clear out connection caches in full when prepping parallel-execution subprocesses. This avoids corner cases causing hangs/freezes due to client/socket reuse. Thanks to Ruslan Lutsenko for the initial report and Romain Chossart for the suggested fix.
- #1026: Fix a typo preventing quiet operation of `fabric.contrib.files.is_link`. Caught by `@dongweiming`.
- #1059: Update IPv6 support to work with link-local address formats. Fix courtesy of `@obormot`.
- #1096: Encode Unicode text appropriately for its target stream object to avoid issues on non-ASCII systems. Thanks to Toru Uetani for the original patch.
- #852: Fix to respect `template_dir` for non Jinja2 templates in `fabric.contrib.files.upload_template`. Thanks to Adam Kowalski for the patch and Alex Plugaru for the initial test case.
- #1134: Skip bad hosts when the tasks are executed in parallel. Thanks to Igor Maravić `@i-maravic`.
- #1146: Fix a bug where `fabric.contrib.files.upload_template` failed to honor `lcd` when `mirror_local_mode` is `True`. Thanks to Laszlo Marai for catch & patch.
- #1147: Use `stat` instead of `lstat` when testing directory-ness in the SFTP module. This allows recursive downloads to avoid recursing into symlinks unexpectedly. Thanks to Igor Kalnitsky for the patch.
- #1165: Prevent infinite loop condition when a gateway host is enabled & the same host is in the regular target host list. Thanks to `@CzBiX` for catch & patch.
- #1167: Add Jinja to `test_requires` in `setup.py` for the couple of newish tests that now require it. Thanks to Kubilay Kocak for the catch.
- #600: Clear out connection caches in full when prepping parallel-execution subprocesses. This avoids corner cases causing hangs/freezes due to client/socket reuse. Thanks to Ruslan Lutsenko for the initial report and Romain Chossart for the suggested fix.
- #1026: Fix a typo preventing quiet operation of `fabric.contrib.files.is_link`. Caught by `@dongweiming`.
- #1059: Update IPv6 support to work with link-local address formats. Fix courtesy of `@obormot`.
- #1096: Encode Unicode text appropriately for its target stream object to avoid issues on non-ASCII systems. Thanks to Toru Uetani for the original patch.
- #852: Fix to respect `template_dir` for non Jinja2 templates in `fabric.contrib.files.upload_template`. Thanks to Adam Kowalski for the patch and Alex Plugaru for the initial test case.
- #1134: Skip bad hosts when the tasks are executed in parallel. Thanks to Igor Maravić `@i-maravic`.
- #1146: Fix a bug where `fabric.contrib.files.upload_template` failed to honor `lcd` when `mirror_local_mode` is `True`. Thanks to Laszlo Marai for catch & patch.
- #1147: Use `stat` instead of `lstat` when testing directory-ness in the SFTP module. This allows recursive downloads to avoid recursing into symlinks unexpectedly. Thanks to Igor Kalnitsky for the patch.
- #1165: Prevent infinite loop condition when a gateway host is enabled & the same host is in the regular target host list. Thanks to `@CzBiX` for catch & patch.
- #741: Add `env.prompts` dictionary, allowing users to set up custom prompt responses (similar to the built-in sudo prompt auto-responder.) Thanks to Nigel Owens and David Halter for the patch.
- #1082: Add `pty` passthrough kwarg to `fabric.contrib.files.upload_template`.
- #1101: Reboot operation now supports custom command. Thanks to Jonas Lejon.
- #938: Add an env var `env.effective_roles` specifying roles used in the currently executing command. Thanks to Piotr Betkier for the patch.

- #1078: Add `.command` and `.real_command` attributes to `local` return value. Thanks to Alexander Teves (`@alexanderteves`) and Konrad Hałas (`@konradhalas`).
- #965: Tweak IO flushing behavior when in linewise (& thus parallel) mode so interwoven output is less frequent. Thanks to `@akidata` for catch & patch.
- : Modified packaging data to reflect that Fabric requires Paramiko < 1.13 (which dropped Python 2.5 support.)
- #1105: Enhance `setup.py` to allow Paramiko 1.13+ under Python 2.6+. Thanks to to `@Arfrever` for catch & patch.
- #1106: Fix a misleading/ambiguous example snippet in the `fab` usage docs to be clearer. Thanks to `@zed`.
- #898: Treat paths that begin with tilde "~" as absolute paths instead of relative. Thanks to Alex Plugaru for the patch and Dan Craig for the suggestion.
- #1105: Enhance `setup.py` to allow Paramiko 1.13+ under Python 2.6+. Thanks to to `@Arfrever` for catch & patch.
- #898: Treat paths that begin with tilde "~" as absolute paths instead of relative. Thanks to Alex Plugaru for the patch and Dan Craig for the suggestion.
- #1105: Enhance `setup.py` to allow Paramiko 1.13+ under Python 2.6+. Thanks to to `@Arfrever` for catch & patch.
- : Modified packaging data to reflect that Fabric requires Paramiko < 1.13 (which dropped Python 2.5 support.)
- : Modified packaging data to reflect that Fabric requires Paramiko < 1.13 (which dropped Python 2.5 support.)
- #1046: Fix typo preventing use of ProxyCommand in some situations. Thanks to Keith Yang.
- #917: Correct an issue with `put(use_sudo=True, mode=xxx)` where the `chmod` was trying to apply to the wrong location. Thanks to Remco (`@nl5887`) for catch & patch.
- #955: Quote directories created as part of `put`'s recursive directory uploads when `use_sudo=True` so directories with shell meta-characters (such as spaces) work correctly. Thanks to John Harris for the catch.
- #1046: Fix typo preventing use of ProxyCommand in some situations. Thanks to Keith Yang.
- #917: Correct an issue with `put(use_sudo=True, mode=xxx)` where the `chmod` was trying to apply to the wrong location. Thanks to Remco (`@nl5887`) for catch & patch.
- #955: Quote directories created as part of `put`'s recursive directory uploads when `use_sudo=True` so directories with shell meta-characters (such as spaces) work correctly. Thanks to John Harris for the catch.
- #948: Handle connection failures due to server load and try connecting to hosts a number of times specified in `env.connection_attempts`.
- #957: Fix bug preventing use of `env.gateway` with targets requiring password authentication. Thanks to Daniel González, `@Bengrunt` and `@adrianbn` for their bug reports.
- #956: Fix pty size detection when running inside Emacs. Thanks to `@akitada` for catch & patch.
- #948: Handle connection failures due to server load and try connecting to hosts a number of times specified in `env.connection_attempts`.
- #957: Fix bug preventing use of `env.gateway` with targets requiring password authentication. Thanks to Daniel González, `@Bengrunt` and `@adrianbn` for their bug reports.
- #956: Fix pty size detection when running inside Emacs. Thanks to `@akitada` for catch & patch.
- #984: Make this changelog easier to read! Now with per-release sections, generated automatically from the old timeline source format.
- #956: Fix pty size detection when running inside Emacs. Thanks to `@akitada` for catch & patch.

- #957: Fix bug preventing use of `env.gateway` with targets requiring password authentication. Thanks to Daniel González, `@Bengrunt` and `@adrianbn` for their bug reports.
- #956: Fix pty size detection when running inside Emacs. Thanks to `@akitada` for catch & patch.
- #957: Fix bug preventing use of `env.gateway` with targets requiring password authentication. Thanks to Daniel González, `@Bengrunt` and `@adrianbn` for their bug reports.
- #910: Added a keyword argument to rsync_project to configure the default options. Thanks to `@moorepants` for the patch.
- #931: Allow overriding of `abort` behavior via a custom exception-returning callable set as `env.abort_exception`. Thanks to Chris Rose for the patch.
- #984: Make this changelog easier to read! Now with per-release sections, generated automatically from the old timeline source format.
- #845: Downstream synchronization option implemented for `fabric.contrib.project.rsync_project`. Thanks to Antonio Barrero for the patch.
- #812: Add `use_glob` option to `put` so users trying to upload real filenames containing glob patterns (`*`, `[` etc) can disable the default globbing behavior. Thanks to Michael McHugh for the patch.
- #826: Enable sudo extraction of compressed archive via `use_sudo` kwarg in `upload_project`. Thanks to `@abec` for the patch.
- #908: Support loading SSH keys from memory. Thanks to Caleb Groom for the patch.
- #924: Add new env var option `colorize-errors` to enable coloring errors and warnings. Thanks to Aaron Meurer for the patch.
- #922: Task argument strings are now displayed when using `fab -d`. Thanks to Kevin Qiu for the patch.
- #925: Added `contrib.files.is_link`. Thanks to `@jtangas` for the patch.
- #84: Fixed problem with missing -r flag in Mac OS X sed version. Thanks to Konrad Hałas for the patch.
- #864: Allow users to disable Fabric's auto-escaping in `run`/`sudo`. Thanks to Christian Long and Michael McHugh for the patch.
- #694: Allow users to work around ownership issues in the default remote login directory: add `temp_dir` kwarg for explicit specification of which "bounce" folder to use when calling `put` with `use_sudo=True`. Thanks to Devin Bayer for the report & Dieter Plaetinck / Jesse Myers for suggesting the workaround.
- #882: Fix a `get` bug regarding spaces in remote working directory names. Thanks to Chris Rose for catch & patch.
- #884: The password cache feature was not working correctly with password-requiring SSH gateway connections. That's fixed now. Thanks to Marco Nenciarini for the catch.
- #171: Added missing cross-references from `env` variables documentation to corresponding command-line options. Thanks to Daniel D. Beck for the contribution.
- #593: Non-ASCII character sets in Jinja templates rendered within `upload_template` would cause `UnicodeDecodeError` when uploaded. This has been addressed by encoding as `utf-8` prior to upload. Thanks to Sébastien Fievet for the catch.
- #912: Leaving `template_dir` un-specified when using `upload_template` in Jinja mode used to cause `'NoneType' has no attribute 'startswith'` errors. This has been fixed. Thanks to Erick Yellott for catch & to Erick Yellott + Kevin Williams for patches.
- #845: Downstream synchronization option implemented for `fabric.contrib.project.rsync_project`. Thanks to Antonio Barrero for the patch.
- #367: Expand paths with tilde inside (`contrib.files`). Thanks to Konrad Hałas for catch & patch.

- #861: Gracefully handle situations where users give a single string literal to `env.hosts`. Thanks to Bill Tucker for catch & patch.
- #871: Use of string mode values in `put(local, remote, mode="NNNN")` would sometimes cause `Unsupported operand` errors. This has been fixed.
- #870: Changes to shell env var escaping highlighted some extraneous and now damaging whitespace in `with path():`. This has been removed and a regression test added.
- #328: `lcd` was no longer being correctly applied to `upload_template`; this has been fixed. Thanks to Joseph Lawson for the catch.
- #868: Substantial speedup of parallel tasks by removing an unnecessary blocking timeout in the `JobQueue` loop. Thanks to Simo Kinnunen for the patch.
- #249: Allow specification of remote command timeout value by setting `env.command_timeout`. Thanks to Paul McMillan for suggestion & initial patch.
- #787: Utilize new Paramiko feature allowing us to skip the use of temporary local files when using file-like objects in `fabric.operations.get`/`fabric.operations.put`.
- #735: Add `ok_ret_codes` option to `env` to allow alternate return codes to be treated os "ok". Thanks to Andy Kraut for the pull request.
- #706: Added `env.tasks`, returning list of tasks to be executed by current `fab` command.
- #818: Added `env.eagerly_disconnect` option to help prevent pile-up of many open connections.
- #730: Add `env.system_known_hosts`/`--system-known-hosts` to allow loading a user-specified system-level SSH `known_hosts` file. Thanks to Roy Smith for the patch.
- #402: Attempt to detect stale SSH sessions and reconnect when they arise. Thanks to `@webengineer` for the patch.
- #823: Add `env.remote_interrupt` which controls whether Ctrl-C is forwarded to the remote end or is captured locally (previously, only the latter behavior was implemented). Thanks to Geert Jansen for the patch.
- #821: Add `fabric.context_managers.remote_tunnel` to allow reverse SSH tunneling (exposing locally-visible network ports to the remote end). Thanks to Giovanni Bajo for the patch.
- #703: Add a `shell` kwarg to many methods in `fabric.contrib.files` to help avoid conflicts with `fabric.context_managers.cd` and similar. Thanks to `@mikek` for the patch.
- #587: Warn instead of aborting when `env.use_ssh_config` is True but the configured SSH conf file doesn't exist. This allows multi-user fabfiles to enable SSH config without causing hard stops for users lacking SSH configs. Thanks to Rodrigo Pimentel for the report.
- #839: Fix bug in `fabric.contrib.project.rsync_project` where IPv6 address were not always correctly detected. Thanks to Antonio Barrero for catch & patch.
- #843: Ensure string `pool_size` values get run through `int()` before deriving final result (stdlib `min()` has odd behavior here...). Thanks to Chris Kastorff for the catch.
- #844: Account for SSH config overhaul in Paramiko 1.10 by e.g. updating treatment of `IdentityFile` to handle multiple values. **This and related SSH config parsing changes are backwards incompatible**; we are including them in this release because they do fix incorrect, off-spec behavior.
- #791: Cast `fabric.operations.reboot`'s `wait` parameter to a numeric type in case the caller submitted a string by mistake. Thanks to Thomas Schreiber for the patch.
- #654: Parallel runs whose sum total of returned data was large (e.g. large return values from the task, or simply a large number of hosts in the host list) were causing frustrating hangs. This has been fixed.

- #805: Update `fabric.context_managers.shell_env` to play nice with Windows (7, at least) systems and `fabric.operations.local`. Thanks to Fernando Macedo for the patch.
- #806: Force strings given to `getpass` during password prompts to be ASCII, to prevent issues on some platforms when Unicode is encountered. Thanks to Alex Louden for the patch.
- : Added current host string to prompt abort error messages.
- #775: Shell escaping was incorrectly applied to the value of `$PATH` updates in our shell environment handling, causing (at the very least) `fabric.operations.local` binary paths to become inoperable in certain situations. This has been fixed.
- #792: The newish `fabric.context_managers.shell_env` context manager was incorrectly omitted from the `fabric.api` import endpoint. This has been remedied. Thanks to Vishal Rana for the catch.
- #604: Fixed wrong treatment of backslashes in put operation when uploading directory tree on Windows. Thanks to Jason Coombs for the catch and `@diresys` & Oliver Janik for the patch. for the patch.
- #766: Use the variable name of a new-style `fabric.tasks.Task` subclass object when the object name attribute is undefined. Thanks to `@todddeluca` for the patch.
- #771: Sphinx autodoc helper `fabric.docs.unwrap_tasks` didn't play nice with `@task(name=xxx)` in some situations. This has been fixed.
- #776: Fixed serious-but-non-obvious bug in direct-tcpip driven gatewaying (e.g. that triggered by `-g` or `env.gateway`.) Should work correctly now.
- #615: Updated `fabric.operations.sudo` to honor the new setting `env.sudo_user` as a default for its `user` kwarg.
- #633: Allow users to turn off host list deduping by setting `env.dedupe_hosts` to `False`. This enables running the same task multiple times on a single host, which was previously not possible.
- #627: Added convenient `quiet` and `warn_only` keyword arguments to `fabric.operations.run`/`fabric.operations.sudo` which are aliases for `settings(hide('everything'), warn_only=True)` and `settings(warn_only=True)`, respectively. (Also added corresponding context managers.) Useful for remote program calls which are expected to fail and/or whose output doesn't need to be shown to users.
- #646: Allow specification of which local streams to use when `fabric.operations.run`/`fabric.operations.sudo` print the remote stdout/stderr, via e.g. `run("command", stderr=sys.stdout)`.
- #241: Add the command executed as a `.command` attribute to the return value of `fabric.operations.run`/`fabric.operations.sudo`. (Also includes a second attribute containing the "real" command executed, including the shell wrapper and any escaping.)
- #669: Updates to our Windows compatibility to rely more heavily on cross-platform Python stdlib implementations. Thanks to Alexey Diyan for the patch.
- #263: Shell environment variable support for `fabric.operations.run`/`fabric.operations.sudo` added in the form of the `fabric.context_managers.shell_env` context manager. Thanks to Oliver Tonnhofer for the original pull request, and to Kamil Kisiel for the final implementation.
- #699: Allow `name` attribute on file-like objects for get/put. Thanks to Peter Lyons for the pull request.
- #491: (also #385:) IPv6 host string support. Thanks to Max Arnold for the patch.
- #725: Updated `fabric.operations.local` to allow override of which local shell is used. Thanks to Mustafa Khattab.
- #723: Add the `group=` argument to `fabric.operations.sudo`. Thanks to Antti Kaihola for the pull request.
- #761: Allow advanced users to parameterize `fabric.main.main()` to force loading of specific fabfiles.

- #578: Add `name` argument to `fabric.decorators.task` to allow overriding of the default "function name is task name" behavior. Thanks to Daniel Simmons for catch & patch.
- #665: (and #629) Update `fabric.contrib.files.upload_template` to have a more useful return value, namely that of its internal `fabric.operations.put` call. Thanks to Miquel Torres for the catch & Rodrigue Alcazar for the patch.
- #763: Add `--initial-password-prompt` to allow prefilling the password cache at the start of a run. Great for sudo-powered parallel runs.
- #684: (also #569) Update how `fabric.decorators.task` wraps task functions to preserve additional metadata; this allows decorated functions to play nice with Sphinx autodoc. Thanks to Jaka Hudoklin for catch & patch.
- #38: (also #698) Implement both SSH-level and `ProxyCommand`-based gatewaying for SSH traffic. (This is distinct from tunneling non-SSH traffic over the SSH connection, which is #78 and not implemented yet.)
  - Thanks in no particular order to Erwin Bolwidt, Oskari Saarenmaa, Steven Noonan, Vladimir Lazarenko, Lincoln de Sousa, Valentino Volonghi, Olle Lundberg and Github user `@acrish` for providing the original patches to both Fabric and Paramiko.
- #702: `fabric.operations.require` failed to test for "empty" values in the env keys it checks (e.g. `require('a-key-whose-value-is-an-empty-list')` would register a successful result instead of alerting that the value was in fact empty. This has been fixed, thanks to Rich Schumacher.
- #711: `fabric.sftp.get` would fail when filenames had % in their path. Thanks to John Begeman
- #704: Fix up a bunch of Python 2.x style `print` statements to be forwards compatible. Thanks to Francesco Del Degan for the patch.
- #736: Ensure context managers that build env vars play nice with `contextlib.nested` by deferring env var reference to entry time, not call time. Thanks to Matthew Tretter for catch & patch.
- #767: Fix (and add test for) regression re: having linewise output automatically activate when parallelism is in effect. Thanks to Alexander Fortin and Dustin McQuay for the bug reports.
- #626: Clarity updates to the tutorial. Thanks to GitHub user `m4z` for the patches.
- #634: Clarified that `fabric.context_managers.lcd` does no special handling re: the user's current working directory, and thus relative paths given to it will be relative to `os.getcwd()`. Thanks to @techtonik for the catch.
- #640: (also #644) Update packaging manifest so sdist tarballs include all necessary test & doc files. Thanks to Mike Gilbert and `@Arfrever` for catch & patch.
- #645: Update Sphinx docs to work well when run out of a source tarball as opposed to a Git checkout. Thanks again to `@Arfrever` for the catch.
- #651: Added note about nesting `with` statements on Python 2.6+. Thanks to Jens Rantil for the patch.
- #681: Fixed outdated docstring for `fabric.decorators.runs_once` which claimed it would get run multiple times in parallel mode. That behavior was fixed in an earlier release but the docs were not updated. Thanks to Jan Brauer for the catch.
- #103: (via #748) Long standing Sphinx autodoc issue requiring error-prone duplication of function signatures in our API docs has been fixed. Thanks to Alex Morega for the patch.
- #684: (also #569) Update how `fabric.decorators.task` wraps task functions to preserve additional metadata; this allows decorated functions to play nice with Sphinx autodoc. Thanks to Jaka Hudoklin for catch & patch.
- #693: Fixed edge case where `abort` driven failures within parallel tasks could result in a top level exception (a `KeyError`) regarding error handling. Thanks to Marcin Kuźmiński for the report.

- #718: `isinstance(foo, Bar)` is used in `fabric.main` instead of `type(foo) == Bar` in order to fix some edge cases. Thanks to Mikhail Korobov.
- #749: Gracefully work around calls to `fabric.version` on systems lacking `/bin/sh` (which causes an `OSError` in `subprocess.Popen` calls.)
- #681: Fixed outdated docstring for `fabric.decorators.runs_once` which claimed it would get run multiple times in parallel mode. That behavior was fixed in an earlier release but the docs were not updated. Thanks to Jan Brauer for the catch.
- #649: Don't swallow non-`abort`-driven exceptions in parallel mode. Fabric correctly printed such exceptions, and returned them from `fabric.tasks.execute`, but did not actually cause the child or parent processes to halt with a nonzero status. This has been fixed. `fabric.tasks.execute` now also honors `env.warn_only` so users may still opt to call it by hand and inspect the returned exceptions, instead of encountering a hard stop. Thanks to Matt Robenolt for the catch.
- #652: Show available commands when aborting on invalid command names.
- #659: Update docs to reflect that `fabric.operations.local` currently honors `env.path`. Thanks to @floledermann for the catch.
- #671: `reject-unknown-hosts` sometimes resulted in a password prompt instead of an abort. This has been fixed. Thanks to Roy Smith for the report.
- #634: Clarified that `fabric.context_managers.lcd` does no special handling re: the user's current working directory, and thus relative paths given to it will be relative to `os.getcwd()`. Thanks to @techtonik for the catch.
- #640: (also #644) Update packaging manifest so sdist tarballs include all necessary test & doc files. Thanks to Mike Gilbert and `@Arfrever` for catch & patch.
- #645: Update Sphinx docs to work well when run out of a source tarball as opposed to a Git checkout. Thanks again to `@Arfrever` for the catch.
- #651: Added note about nesting `with` statements on Python 2.6+. Thanks to Jens Rantil for the patch.

- #649: Don't swallow non-`abort`-driven exceptions in parallel mode. Fabric correctly printed such exceptions, and returned them from `fabric.tasks.execute`, but did not actually cause the child or parent processes to halt with a nonzero status. This has been fixed. `fabric.tasks.execute` now also honors `env.warn_only` so users may still opt to call it by hand and inspect the returned exceptions, instead of encountering a hard stop. Thanks to Matt Robenolt for the catch.
- #652: Show available commands when aborting on invalid command names.
- #659: Update docs to reflect that `fabric.operations.local` currently honors `env.path`. Thanks to @floledermann for the catch.
- #671: `reject-unknown-hosts` sometimes resulted in a password prompt instead of an abort. This has been fixed. Thanks to Roy Smith for the report.
- #634: Clarified that `fabric.context_managers.lcd` does no special handling re: the user's current working directory, and thus relative paths given to it will be relative to `os.getcwd()`. Thanks to @techtonik for the catch.
- #640: (also #644) Update packaging manifest so sdist tarballs include all necessary test & doc files. Thanks to Mike Gilbert and `@Arfrever` for catch & patch.
- #645: Update Sphinx docs to work well when run out of a source tarball as opposed to a Git checkout. Thanks again to `@Arfrever` for the catch.
- #651: Added note about nesting `with` statements on Python 2.6+. Thanks to Jens Rantil for the patch.

- #610: Change detection of `env.key_filename`'s type (added as part of SSH config support in 1.4) so it supports arbitrary iterables. Thanks to Brandon Rhodes for the catch.
- #609: (and #564) Document and clean up `env.sudo_prefix` so it can be more easily modified by users facing uncommon use cases. Thanks to GitHub users `3point2` for the cleanup and `SirScott` for the documentation catch.
- #616: Add port number to the error message displayed upon connection failures.
- #617: Fix the `clean_revert` behavior of `fabric.context_managers.settings` so it doesn't `KeyError` for newly created settings keys. Thanks to Chris Streeter for the catch.
- #624: Login password prompts did not always display the username being authenticated for. This has been fixed. Thanks to Nick Zalutskiy for catch & patch.
- #625: `fabric.context_managers.hide`/`fabric.context_managers.show` did not correctly restore prior display settings if an exception was raised inside the block. This has been fixed.
- #562: Agent forwarding would error out or freeze when multiple uses of the forwarded agent were used per remote invocation (e.g. a single `fabric.operations.run` command resulting in multiple Git or SVN checkouts.) This has been fixed thanks to Steven McDonald and GitHub user `@lynxis`.
- #626: Clarity updates to the tutorial. Thanks to GitHub user `m4z` for the patches.
- #610: Change detection of `env.key_filename`'s type (added as part of SSH config support in 1.4) so it supports arbitrary iterables. Thanks to Brandon Rhodes for the catch.
- #609: (and #564) Document and clean up `env.sudo_prefix` so it can be more easily modified by users facing uncommon use cases. Thanks to GitHub users `3point2` for the cleanup and `SirScott` for the documentation catch.
- #616: Add port number to the error message displayed upon connection failures.
- #617: Fix the `clean_revert` behavior of `fabric.context_managers.settings` so it doesn't `KeyError` for newly created settings keys. Thanks to Chris Streeter for the catch.
- #624: Login password prompts did not always display the username being authenticated for. This has been fixed. Thanks to Nick Zalutskiy for catch & patch.
- #625: `fabric.context_managers.hide`/`fabric.context_managers.show` did not correctly restore prior display settings if an exception was raised inside the block. This has been fixed.
- #562: Agent forwarding would error out or freeze when multiple uses of the forwarded agent were used per remote invocation (e.g. a single `fabric.operations.run` command resulting in multiple Git or SVN checkouts.) This has been fixed thanks to Steven McDonald and GitHub user `@lynxis`.
- #626: Clarity updates to the tutorial. Thanks to GitHub user `m4z` for the patches.
- #306: Remote paths now use posixpath for a separator. Thanks to Jason Coombs for the patch.
- #572: Parallel task aborts (as oppposed to unhandled exceptions) now correctly print their abort messages instead of tracebacks, and cause the parent process to exit with the correct (nonzero) return code. Thanks to Ian Langworth for the catch.
- #551: `--list` output now detects terminal window size and truncates (or doesn't truncate) accordingly. Thanks to Horacio G. de Oro for the initial pull request.
- #499: `contrib.files.first` used an outdated function signature in its wrapped `fabric.contrib.files.exists` call. This has been fixed. Thanks to Massimiliano Torromeo for catch & patch.
- #458: `fabric.decorators.with_settings` did not perfectly match `fabric.context_managers.settings`, re: ability to inline additional context managers. This has been corrected. Thanks to Rory Geoghegan for the patch.

- #584: `fabric.contrib.project.upload_project` did not take explicit remote directory location into account when untarring, and now uses `fabric.context_managers.cd` to address this. Thanks to Ben Burry for the patch.
- #568: `fabric.tasks.execute` allowed too much of its internal state changes (to variables such as `env.host_string` and `env.parallel`) to persist after execution completed; this caused a number of different incorrect behaviors. `fabric.tasks.execute` has been overhauled to clean up its own state changes – while preserving any state changes made by the task being executed.
- #395: Added an FAQ entry detailing how to handle init scripts which misbehave when a pseudo-tty is allocated.
- #607: Allow `fabric.operations.local` to display stdout/stderr when it warns/aborts, if it was capturing them.
- #608: Add `capture` kwarg to `fabric.contrib.project.rsync_project` to aid in debugging rsync problems.
- #306: Remote paths now use posixpath for a separator. Thanks to Jason Coombs for the patch.
- #572: Parallel task aborts (as oppposed to unhandled exceptions) now correctly print their abort messages instead of tracebacks, and cause the parent process to exit with the correct (nonzero) return code. Thanks to Ian Langworth for the catch.
- #551: `--list` output now detects terminal window size and truncates (or doesn't truncate) accordingly. Thanks to Horacio G. de Oro for the initial pull request.
- #499: `contrib.files.first` used an outdated function signature in its wrapped `fabric.contrib.files.exists` call. This has been fixed. Thanks to Massimiliano Torromeo for catch & patch.
- #458: `fabric.decorators.with_settings` did not perfectly match `fabric.context_managers.settings`, re: ability to inline additional context managers. This has been corrected. Thanks to Rory Geoghegan for the patch.
- #584: `fabric.contrib.project.upload_project` did not take explicit remote directory location into account when untarring, and now uses `fabric.context_managers.cd` to address this. Thanks to Ben Burry for the patch.
- #568: `fabric.tasks.execute` allowed too much of its internal state changes (to variables such as `env.host_string` and `env.parallel`) to persist after execution completed; this caused a number of different incorrect behaviors. `fabric.tasks.execute` has been overhauled to clean up its own state changes – while preserving any state changes made by the task being executed.
- #395: Added an FAQ entry detailing how to handle init scripts which misbehave when a pseudo-tty is allocated.
- #607: Allow `fabric.operations.local` to display stdout/stderr when it warns/aborts, if it was capturing them.
- #608: Add `capture` kwarg to `fabric.contrib.project.rsync_project` to aid in debugging rsync problems.
- #72: SSH agent forwarding support has made it into Fabric's SSH library, and hooks for using it have been added (disabled by default; use `-A` or `env.forward_agent` to enable.) Thanks to Ben Davis for porting an existing Paramiko patch to `ssh` and providing the necessary tweak to Fabric.
- #506: A new output alias, `commands`, has been added, which allows hiding remote stdout and local "running command X" output lines.
- #13: Env vars may now be set at runtime via the new `--set` command-line flag.
- #8: Added `--skip-bad-hosts`/`env.skip_bad_hosts` option to allow skipping past temporarily down/unreachable hosts.

- #474: `fabric.tasks.execute` now allows you to access the executed task's return values, by itself returning a dictionary whose keys are the host strings executed against.
- #12: Added the ability to try connecting multiple times to temporarily-down remote systems, instead of immediately failing. (Default behavior is still to only try once.) See `env.timeout` and `env.connection_attempts` for controlling both connection timeouts and total number of attempts. `fabric.operations.reboot` has also been overhauled (but practically deprecated – see its updated docs.)
- #3: Fabric can now load a subset of SSH config functionality directly from your local `~/.ssh/config` if `env.use_ssh_config` is set to `True`. See `ssh-config` for details. Thanks to Kirill Pinchuk for the initial patch.
- #138: `env.port` may now be written to at fabfile module level to set a default nonstandard port number. Previously this value was read-only.
- #559: `fabric.contrib.project.rsync_project` now allows users to append extra SSH-specific arguments to `rsync`'s `--rsh` flag.
- #487: Overhauled the regular expression escaping performed in `fabric.contrib.files.append` and `fabric.contrib.files.contains` to try and handle more corner cases. Thanks to Neilen Marais for the patch.
- #467: (also #468, #469) Handful of documentation clarification tweaks. Thanks to Paul Hoffman for the patches.
- #459: Update our `setup.py` files to note that PyCrypto released 2.4.1, which fixes the setuptools problems.
- #532: Reorganized and cleaned up the output of `fab --help`.
- #410: Fixed a bug where using the `fabric.decorators.task` decorator inside/under another decorator such as `fabric.decorators.hosts` could cause that task to become invalid when invoked by name (due to how old-style vs new-style tasks are detected.) Thanks to Dan Colish for the initial patch.
- #495: Fixed documentation example showing how to subclass `fabric.tasks.Task`. Thanks to Brett Haydon for the catch and Mark Merritt for the patch.
- #339: Don't show imported `fabric.colors` members in `--list` output. Thanks to Nick Trew for the report.
- #494: Fixed regression bug affecting some `env` values such as `env.port` under parallel mode. Symptoms included `fabric.contrib.project.rsync_project` bailing out due to a None port value when run under `@parallel`. Thanks to Rob Terhaar for the report.
- #510: Parallel mode is incompatible with user input, such as password/hostname prompts, and was causing cryptic `Operation not supported by device` errors when such prompts needed to be displayed. This behavior has been updated to cleanly and obviously `abort` instead.
- #492: `@parallel` did not automatically trigger linewise output, as was intended. This has been fixed. Thanks to Brandon Huey for the catch.
- #410: Fixed a bug where using the `fabric.decorators.task` decorator inside/under another decorator such as `fabric.decorators.hosts` could cause that task to become invalid when invoked by name (due to how old-style vs new-style tasks are detected.) Thanks to Dan Colish for the initial patch.
- #495: Fixed documentation example showing how to subclass `fabric.tasks.Task`. Thanks to Brett Haydon for the catch and Mark Merritt for the patch.
- #339: Don't show imported `fabric.colors` members in `--list` output. Thanks to Nick Trew for the report.
- #494: Fixed regression bug affecting some `env` values such as `env.port` under parallel mode. Symptoms included `fabric.contrib.project.rsync_project` bailing out due to a None port value when run under `@parallel`. Thanks to Rob Terhaar for the report.

- #510: Parallel mode is incompatible with user input, such as password/hostname prompts, and was causing cryptic `Operation not supported by device` errors when such prompts needed to be displayed. This behavior has been updated to cleanly and obviously `abort` instead.
- #492: `@parallel` did not automatically trigger linewise output, as was intended. This has been fixed. Thanks to Brandon Huey for the catch.
- #410: Fixed a bug where using the `fabric.decorators.task` decorator inside/under another decorator such as `fabric.decorators.hosts` could cause that task to become invalid when invoked by name (due to how old-style vs new-style tasks are detected.) Thanks to Dan Colish for the initial patch.
- #495: Fixed documentation example showing how to subclass `fabric.tasks.Task`. Thanks to Brett Haydon for the catch and Mark Merritt for the patch.
- #339: Don't show imported `fabric.colors` members in `--list` output. Thanks to Nick Trew for the report.
- #494: Fixed regression bug affecting some `env` values such as `env.port` under parallel mode. Symptoms included `fabric.contrib.project.rsync_project` bailing out due to a None port value when run under `@parallel`. Thanks to Rob Terhaar for the report.
- #510: Parallel mode is incompatible with user input, such as password/hostname prompts, and was causing cryptic `Operation not supported by device` errors when such prompts needed to be displayed. This behavior has been updated to cleanly and obviously `abort` instead.
- #492: `@parallel` did not automatically trigger linewise output, as was intended. This has been fixed. Thanks to Brandon Huey for the catch.
- #230: Fix regression re: combo of no fabfile & arbitrary command use. Thanks to Ali Saifee for the catch.
- #482: Parallel mode should imply linewise output; omission of this behavior was an oversight.
- #342: Combining `fabric.context_managers.cd` with `fabric.operations.put` and its `use_sudo` keyword caused an unrecoverable error. This has been fixed. Thanks to Egor M for the report.
- #341: `fabric.contrib.files.append` incorrectly failed to detect that the line(s) given already existed in files hidden to the remote user, and continued appending every time it ran. This has been fixed. Thanks to Dominique Peretti for the catch and Martin Vilcans for the patch.
- #397: Some poorly behaved objects in third party modules triggered exceptions during Fabric's "classic or new-style task?" test. A fix has been added which tries to work around these.
- #400: Handle corner case of systems where `pwd.getpwuid` raises `KeyError` for the user's UID instead of returning a valid string. Thanks to Dougal Matthews for the catch.
- #437: `fabric.decorators.with_settings` now correctly preserves the wrapped function's docstring and other attributes. Thanks to Eric Buckley for the catch and Luke Plant for the patch.
- #443: `fabric.contrib.files.exists` didn't expand tildes; now it does. Thanks to Riccardo Magliocchetti for the patch.
- #446: Add QNX to list of secondary-case `fabric.contrib.files.sed` targets. Thanks to Rodrigo Madruga for the tip.
- #450: Improve traceback display when handling `ImportError` for dependencies. Thanks to David Wolever for the patches.
- #475: Allow escaping of equals signs in per-task args/kwargs.
- #441: Specifying a task module as a task on the command line no longer blows up but presents the usual "no task by that name" error message instead. Thanks to Mitchell Hashimoto for the catch.
- #230: Fix regression re: combo of no fabfile & arbitrary command use. Thanks to Ali Saifee for the catch.

- #482: Parallel mode should imply linewise output; omission of this behavior was an oversight.
- #342: Combining `fabric.context_managers.cd` with `fabric.operations.put` and its `use_sudo` keyword caused an unrecoverable error. This has been fixed. Thanks to Egor M for the report.
- #341: `fabric.contrib.files.append` incorrectly failed to detect that the line(s) given already existed in files hidden to the remote user, and continued appending every time it ran. This has been fixed. Thanks to Dominique Peretti for the catch and Martin Vilcans for the patch.
- #397: Some poorly behaved objects in third party modules triggered exceptions during Fabric's "classic or new-style task?" test. A fix has been added which tries to work around these.
- #400: Handle corner case of systems where `pwd.getpwuid` raises `KeyError` for the user's UID instead of returning a valid string. Thanks to Dougal Matthews for the catch.
- #437: `fabric.decorators.with_settings` now correctly preserves the wrapped function's docstring and other attributes. Thanks to Eric Buckley for the catch and Luke Plant for the patch.
- #443: `fabric.contrib.files.exists` didn't expand tildes; now it does. Thanks to Riccardo Magliocchetti for the patch.
- #446: Add QNX to list of secondary-case `fabric.contrib.files.sed` targets. Thanks to Rodrigo Madruga for the tip.
- #450: Improve traceback display when handling `ImportError` for dependencies. Thanks to David Wolever for the patches.
- #475: Allow escaping of equals signs in per-task args/kwargs.
- #441: Specifying a task module as a task on the command line no longer blows up but presents the usual "no task by that name" error message instead. Thanks to Mitchell Hashimoto for the catch.
- #230: Fix regression re: combo of no fabfile & arbitrary command use. Thanks to Ali Saifee for the catch.
- #482: Parallel mode should imply linewise output; omission of this behavior was an oversight.
- #342: Combining `fabric.context_managers.cd` with `fabric.operations.put` and its `use_sudo` keyword caused an unrecoverable error. This has been fixed. Thanks to Egor M for the report.
- #341: `fabric.contrib.files.append` incorrectly failed to detect that the line(s) given already existed in files hidden to the remote user, and continued appending every time it ran. This has been fixed. Thanks to Dominique Peretti for the catch and Martin Vilcans for the patch.
- #397: Some poorly behaved objects in third party modules triggered exceptions during Fabric's "classic or new-style task?" test. A fix has been added which tries to work around these.
- #400: Handle corner case of systems where `pwd.getpwuid` raises `KeyError` for the user's UID instead of returning a valid string. Thanks to Dougal Matthews for the catch.
- #437: `fabric.decorators.with_settings` now correctly preserves the wrapped function's docstring and other attributes. Thanks to Eric Buckley for the catch and Luke Plant for the patch.
- #443: `fabric.contrib.files.exists` didn't expand tildes; now it does. Thanks to Riccardo Magliocchetti for the patch.
- #446: Add QNX to list of secondary-case `fabric.contrib.files.sed` targets. Thanks to Rodrigo Madruga for the tip.
- #450: Improve traceback display when handling `ImportError` for dependencies. Thanks to David Wolever for the patches.
- #475: Allow escaping of equals signs in per-task args/kwargs.
- #441: Specifying a task module as a task on the command line no longer blows up but presents the usual "no task by that name" error message instead. Thanks to Mitchell Hashimoto for the catch.

- #467: (also #468, #469) Handful of documentation clarification tweaks. Thanks to Paul Hoffman for the patches.
- #459: Update our `setup.py` files to note that PyCrypto released 2.4.1, which fixes the setuptools problems.
- #457: Ensured that Fabric fast-fails parallel tasks if any child processes encountered errors. Previously, multi-task invocations would continue to the 2nd, etc task when failures occurred, which does not fit with how Fabric usually behaves. Thanks to Github user `sdcooke` for the report and Morgan Goose for the fix.
- #467: (also #468, #469) Handful of documentation clarification tweaks. Thanks to Paul Hoffman for the patches.
- #459: Update our `setup.py` files to note that PyCrypto released 2.4.1, which fixes the setuptools problems.
- #457: Ensured that Fabric fast-fails parallel tasks if any child processes encountered errors. Previously, multi-task invocations would continue to the 2nd, etc task when failures occurred, which does not fit with how Fabric usually behaves. Thanks to Github user `sdcooke` for the report and Morgan Goose for the fix.
- #467: (also #468, #469) Handful of documentation clarification tweaks. Thanks to Paul Hoffman for the patches.
- #459: Update our `setup.py` files to note that PyCrypto released 2.4.1, which fixes the setuptools problems.
- #457: Ensured that Fabric fast-fails parallel tasks if any child processes encountered errors. Previously, multi-task invocations would continue to the 2nd, etc task when failures occurred, which does not fit with how Fabric usually behaves. Thanks to Github user `sdcooke` for the report and Morgan Goose for the fix.
- #19: Tasks may now be optionally executed in parallel. Please see the parallel execution docs for details. Major thanks to Morgan Goose for the initial implementation.
- #21: It is now possible, using the new `fabric.tasks.execute` API call, to execute task objects (by reference or by name) from within other tasks or in library mode. `fabric.tasks.execute` honors the other tasks' `fabric.decorators.hosts`/`fabric.decorators.roles` decorators, and also supports passing in explicit host and/or role arguments.
- #416: Updated documentation to reflect move from Redmine to Github.
- #393: Fixed a typo in an example code snippet in the task docs. Thanks to Hugo Garza for the catch.
- #275: To support an edge use case of the features released in #19, and to lay the foundation for #275, we have forked Paramiko into the Python 'ssh' library and changed our dependency to it for Fabric 1.3 and higher. This may have implications for the more uncommon install use cases, and package maintainers, but we hope to iron out any issues as they come up.
- #430: Tasks decorated with `fabric.decorators.runs_once` printed extraneous 'Executing...' status lines on subsequent invocations. This is noisy at best and misleading at worst, and has been corrected. Thanks to Jacob Kaplan-Moss for the report.
- #182: During display of remote stdout/stderr, Fabric occasionally printed extraneous line prefixes (which in turn sometimes overwrote wrapped text.) This has been fixed.
- #323: `fabric.operations.put` forgot how to expand leading tildes in the remote file path. This has been corrected. Thanks to Piet Delport for the catch.
- #430: Tasks decorated with `fabric.decorators.runs_once` printed extraneous 'Executing...' status lines on subsequent invocations. This is noisy at best and misleading at worst, and has been corrected. Thanks to Jacob Kaplan-Moss for the report.
- #182: During display of remote stdout/stderr, Fabric occasionally printed extraneous line prefixes (which in turn sometimes overwrote wrapped text.) This has been fixed.
- #323: `fabric.operations.put` forgot how to expand leading tildes in the remote file path. This has been corrected. Thanks to Piet Delport for the catch.
- #430: Tasks decorated with `fabric.decorators.runs_once` printed extraneous 'Executing...' status lines on subsequent invocations. This is noisy at best and misleading at worst, and has been corrected. Thanks to Jacob Kaplan-Moss for the report.

- #182: During display of remote stdout/stderr, Fabric occasionally printed extraneous line prefixes (which in turn sometimes overwrote wrapped text.) This has been fixed.
- #323: `fabric.operations.put` forgot how to expand leading tildes in the remote file path. This has been corrected. Thanks to Piet Delport for the catch.
- #303: Updated terminal size detection to correctly skip over non-tty stdout, such as when running `fab taskname | other_command`.
- #373: Re-added missing functionality preventing host exclusion from working correctly.
- #396: `--shortlist` broke after the addition of `--list-format` and no longer displayed the short list format correctly. This has been fixed.
- #252: `fabric.context_managers.settings` would silently fail to set `env` values for keys which did not exist outside the context manager block. It now works as expected. Thanks to Will Maier for the catch and suggested solution.
- #393: Fixed a typo in an example code snippet in the task docs. Thanks to Hugo Garza for the catch.
- #303: Updated terminal size detection to correctly skip over non-tty stdout, such as when running `fab taskname | other_command`.
- #373: Re-added missing functionality preventing host exclusion from working correctly.
- #396: `--shortlist` broke after the addition of `--list-format` and no longer displayed the short list format correctly. This has been fixed.
- #252: `fabric.context_managers.settings` would silently fail to set `env` values for keys which did not exist outside the context manager block. It now works as expected. Thanks to Will Maier for the catch and suggested solution.
- #393: Fixed a typo in an example code snippet in the task docs. Thanks to Hugo Garza for the catch.
- #303: Updated terminal size detection to correctly skip over non-tty stdout, such as when running `fab taskname | other_command`.
- #373: Re-added missing functionality preventing host exclusion from working correctly.
- #396: `--shortlist` broke after the addition of `--list-format` and no longer displayed the short list format correctly. This has been fixed.
- #252: `fabric.context_managers.settings` would silently fail to set `env` values for keys which did not exist outside the context manager block. It now works as expected. Thanks to Will Maier for the catch and suggested solution.
- #393: Fixed a typo in an example code snippet in the task docs. Thanks to Hugo Garza for the catch.
- #389: Fixed/improved error handling when Paramiko import fails. Thanks to Brian Luft for the catch.
- #417: `abort-on-prompts` would incorrectly abort when set to True, even if both password and host were defined. This has been fixed. Thanks to Valerie Ishida for the report.
- #416: Updated documentation to reflect move from Redmine to Github.
- #380: Improved unicode support when testing objects for being string-like. Thanks to Jiri Barton for catch & patch.
- #389: Fixed/improved error handling when Paramiko import fails. Thanks to Brian Luft for the catch.
- #417: `abort-on-prompts` would incorrectly abort when set to True, even if both password and host were defined. This has been fixed. Thanks to Valerie Ishida for the report.
- #416: Updated documentation to reflect move from Redmine to Github.

- #380: Improved unicode support when testing objects for being string-like. Thanks to Jiri Barton for catch & patch.
- #389: Fixed/improved error handling when Paramiko import fails. Thanks to Brian Luft for the catch.
- #417: `abort-on-prompts` would incorrectly abort when set to True, even if both password and host were defined. This has been fixed. Thanks to Valerie Ishida for the report.
- #416: Updated documentation to reflect move from Redmine to Github.
- #22: Enhanced `@task` to add aliasing, per-module default tasks, and control over the wrapping task class. Thanks to Travis Swicegood for the initial work and collaboration.
- #382: Experimental overhaul of changelog formatting & process to make supporting multiple lines of development less of a hassle.

## 4.3 Frequently Asked/Answered Questions (FAQ)

These are some of the most commonly encountered problems or frequently asked questions which we receive from users. They aren't intended as a substitute for reading the rest of the documentation, so please make sure you check it out if your question is not answered here.

---

**Note:** Most API examples and links are for version 2 and up; FAQs specific to version 1 will typically be marked as such.

---

**Warning:** Many questions about shell command execution and task behavior are answered on Invoke's FAQ page - please check there also!

### 4.3.1 Explicitly set env variables are not being set correctly on the remote end!

If your attempts to set environment variables for things like `Connection.run` appear to silently fail, you're almost certainly talking to an SSH server which is setting a highly restrictive AcceptEnv.

To fix, you can either modify the server's configuration to allow the env vars you're setting, or use the `inline_ssh_env` `Connection` parameter (or the global config option of the same name) to force Fabric to send env vars prefixed before your command strings instead.

### 4.3.2 The remote shell environment doesn't match interactive shells!

You may find environment variables (or the behavior they trigger) differ interactively vs scripted via Fabric. For example, a program that's on your `$PATH` when you manually `ssh` in might not be visible when using `Connection.run`; or special per-program env vars such as those for Python, pip, Java etc are not taking effect; etc.

The root cause of this is typically because the SSH server runs non-interactive commands via a very limited shell call: `/path/to/shell -c "command"` (for example, OpenSSH). Most shells, when run this way, are not considered to be either **interactive** or **login** shells; and this then impacts which startup files get loaded.

Users typically only modify shell files related to interactive operation (such as `~/.bash_profile` or `/etc/zshrc`); such changes do not take effect when the SSH server is running one-off commands.

To work around this, consult your shell's documentation to see if it offers any non-login, non-interactive config files; for example, `zsh` lets you configure `/etc/zshrc` or `~/.zshenv` for this purpose.

**Note:** `bash` does not appear to offer standard non-login/non-interactive startup files, even in version 4. However, it may attempt to determine if it's being run by a remote-execution daemon and will apparently source `~/.bashrc` if so; check to see if this is the case on your target systems.

**Note:** Another workaround for `bash` users is to reply on its `$BASH_ENV` functionality, which names a file path as the startup file to load:

- configure your SSH server to `AcceptEnv BASH_ENV`, so that you can actually set that env var for the remote session at the top level (most SSH servers disallow this method by default).
- decide which file this should be, though if you're already modifying files like `~/.bash_profile` or `~/.bashrc`, you may want to just point at that exact path.
- set the Fabric configuration value `run.env` to aim at the above path, e.g. `{"BASH_ENV": "~/.bash_profile"}`.

### 4.3.3 My (`cd`/`workon`/`export`/etc) calls don't seem to work!

While Fabric can be used for many shell-script-like tasks, there's a slightly unintuitive catch: each `run` or `sudo` call (or the `run`/`sudo` functions in v1) has its own distinct shell session. This is required in order for Fabric to reliably figure out, after your command has run, what its standard out/error and return codes were.

Unfortunately, it means that code like the following doesn't behave as you might assume:

```
@task
def deploy(c):
    c.run("cd /path/to/application")
    c.run("./update.sh")
```

If that were a shell script, the second `run` call would have executed with a current working directory of `/path/to/application/` – but because both commands are run in their own distinct session over SSH, it actually tries to execute `$HOME/update.sh` instead (since your remote home directory is the default working directory).

A simple workaround is to make use of shell logic operations such as `&&`, which link multiple expressions together (provided the left hand side executed without error) like so:

```
def deploy(c):
    c.run("cd /path/to/application && ./update.sh")
```

**Note:** You might also get away with an absolute path and skip directory changing altogether:

```
def deploy(c):
    c.run("/path/to/application/update.sh")
```

However, this requires that the command in question makes no assumptions about your current working directory!

### 4.3.4 Why do I sometimes see `err: stdin: is not a tty`?

See Invoke's FAQ for this; even for Fabric v1, which is not based on Invoke, the answer is the same.

### 4.3.5 Why can't I run programs in the background with `&`? It makes Fabric hang.

Because SSH executes a new shell session on the remote end for each invocation of `run` or `sudo` (*see also*), backgrounded processes may prevent the calling shell from exiting until the processes stop running, which in turn prevents Fabric from continuing on with its own execution.

The key to fixing this is to ensure that your process' standard pipes are all disassociated from the calling shell, which may be done in a number of ways (listed in order of robustness):

- Use a pre-existing daemonization technique if one exists for the program at hand – for example, calling an init script instead of directly invoking a server binary.
  - Or leverage a process manager such as `supervisord`, `upstart` or `systemd` - such tools let you define what it means to "run" one of your background processes, then issue init-script-like start/stop/restart/status commands. They offer many advantages over classic init scripts as well.
- Use `tmux`, `screen` or `dtach` to fully detach the process from the running shell; these tools have the benefit of allowing you to reattach to the process later on if needed (though they are more ad-hoc than `supervisord`-like tools).
- Run the program under `nohup` or similar "in-shell" tools - note that this approach has seen limited success for most users.

### 4.3.6 I'm sometimes incorrectly asked for a passphrase instead of a password.

Due to a bug of sorts in our SSH layer, it's not currently possible for Fabric to always accurately detect the type of authentication needed. We have to try and guess whether we're being asked for a private key passphrase or a remote server password, and in some cases our guess ends up being wrong.

The most common such situation is where you, the local user, appear to have an SSH keychain agent running, but the remote server is not able to honor your SSH key, e.g. you haven't yet transferred the public key over or are using an incorrect username. In this situation, Fabric will prompt you with "Please enter passphrase for private key", but the text you enter is actually being sent to the remote end's password authentication.

We hope to address this in future releases by contributing to the aforementioned SSH library.

## 4.4 Installing

**Note:** Users looking to install Fabric 1.x should see *Installing (1.x)*. However, *upgrading* to 2.x is strongly recommended.

Fabric is best installed via pip:

```
$ pip install fabric
```

All advanced `pip` use cases work too, such as:

```
$ pip install -e git+https://github.com/fabric/fabric
```

Or cloning the Git repository and running:

```
$ pip install -e .
```

within it.

Your operating system may also have a Fabric package available (though these are typically older and harder to support), typically called `fabric` or `python-fabric`. E.g.:

```
$ sudo apt-get install fabric
```

### 4.4.1 Installing modern Fabric as `fabric2`

Users who are migrating from Fabric 1 to Fabric 2+ may find it useful to have both versions installed side-by-side. The easiest way to do this is to use the handy `fabric2` PyPI entry:

```
$ pip install fabric2
```

This upload is generated from the normal Fabric repository, but is tweaked at build time so that it installs a `fabric2` package instead of a `fabric` one (and a `fab2` binary instead of a `fab` one.) The codebase is otherwise unchanged.

Users working off of the Git repository can enable that same tweak with an environment variable, e.g.:

```
$ PACKAGE_AS_FABRIC2=yes pip install -e .
```

**Note:** The value of the environment variable doesn't matter, as long as it is not empty.

#### `fabric` and `fabric2` vs `fabric3`

Unfortunately, the `fabric3` entry on PyPI is an unauthorized fork of Fabric 1.x which we do not control. Once modern Fabric gets up to 3.x, 4.x etc, we'll likely continue distributing it via both `fabric` and `fabric2` for convenience; there will never be any official `fabric3`, `fabric4` etc.

In other words, `fabric2` is purely there to help users of 1.x cross the 2.0 "major rewrite" barrier; future major versions will *not* be large rewrites and will only have small sets of backward incompatibilities.

#### Inability to `pip install -e` both versions

You may encounter issues if *both* versions of Fabric are installed via `pip install -e`, due to how that functionality works (tl;dr it just adds the checkout directories to `sys.path`, regardless of whether you wanted to "install" all packages within them - so Fabric 2+'s `fabric/` package still ends up visible to the import system alongside `fabric2/`).

Thus, you may only have one of the local copies of Fabric installed in 'editable' fashion at a time, and the other must be repeatedly reinstalled via `pip install` (no `-e`) if you need to make edits to it.

#### Order of installations

Due to the same pip quirk mentioned above, if either of your Fabric versions are installed in 'editable' mode, you **must** install the 'editable' version first, and then install the 'static' version second.

For example, if you're migrating from some public release of Fabric 1 to a checkout of modern Fabric:

```
$ PACKAGE_AS_FABRIC2=yes pip install -e /path/to/fabric2
$ pip install fabric==1.14.0
```

You may see some warnings on that second `pip install` (eg `Not uninstalling fabric` or `Can't uninstall 'fabric'`) but as long as it exits cleanly and says something like `Successfully installed fabric-1.14.0`, you should be okay. Double check with e.g. `pip list` and you should have entries for both `fabric` and `fabric2`.

### 4.4.2 Dependencies

In order for Fabric's installation to succeed, you will need the following:

- the Python programming language, versions 2.7 or 3.4+;
- the Invoke command-running and task-execution library;
- and the Paramiko SSH library (as well as its own dependencies; see its install docs.)

#### Development dependencies

If you are interested in doing development work on Fabric (or even just running the test suite), you'll need the libraries listed in the `dev-requirements.txt` (included in the source distribution.) Usually it's easy to simply `pip install -r dev-requirements.txt`.

### 4.4.3 Downloads

To obtain a tar.gz or zip archive of the Fabric source code, you may visit Fabric's PyPI page, which offers manual downloads in addition to being the entry point for `pip`.

### 4.4.4 Source code checkouts

The Fabric developers manage the project's source code with the Git DVCS. To follow Fabric's development via Git instead of downloading official releases, you have the following options:

- Clone the canonical repository straight from the Fabric organization's repository on Github (cloning instructions available on that page).
- Make your own fork of the Github repository by making a Github account, visiting fabric/fabric and clicking the "fork" button.

---

**Note:** If you've obtained the Fabric source via source control and plan on updating your checkout in the future, we highly suggest using `pip install -e .` (or `python setup.py develop`) instead – it will use symbolic links instead of file copies, ensuring that imports of the library or use of the command-line tool will always refer to your checkout.

---

For information on the hows and whys of Fabric development, including which branches may be of interest and how you can help out, please see the *Development* page.

## 4.5 Installing (1.x)

---

**Note:** Installing Fabric 2.0 or above? Looking for non-PyPI downloads or source code checkout instructions? See *Installing*.

---

This document includes legacy notes on installing Fabric 1.x. Users are strongly encouraged to upgrade to 2.x when possible.

### 4.5.1 Basic installation

Fabric is best installed via pip; to ensure you get Fabric 1 instead of the new but incompatible Fabric 2, specify `<2.0`:

```
$ pip install 'fabric<2.0'
```

All advanced `pip` use cases work too, such as installing the latest copy of the `v1` development branch:

```
$ pip install -e 'git+https://github.com/fabric/fabric@v1#egg=fabric'
```

Or cloning the Git repository and running:

```
$ git checkout v1
$ pip install -e .
```

within it.

Your operating system may also have a Fabric package available (though these are typically older and harder to support), typically called `fabric` or `python-fabric`. E.g.:

```
$ sudo apt-get install fabric
```

**Note:** Make sure to confirm which major version is currently packaged!

### 4.5.2 Dependencies

In order for Fabric's installation to succeed, you will need four primary pieces of software:

- the Python programming language;
- the `setuptools` packaging/installation library;
- the Python Paramiko SSH library;
- and Paramiko's dependency, Cryptography.

and, if using parallel execution mode,

- the *multiprocessing* library.

Please read on for important details on each dependency – there are a few gotchas.

#### Python

Fabric requires Python version 2.5+.

#### setuptools

Setuptools comes with most Python installations by default; if yours doesn't, you'll need to grab it. In such situations it's typically packaged as `python-setuptools`, `py26-setuptools` or similar.

#### multiprocessing

An optional dependency, the `multiprocessing` library is included in Python's standard library in version 2.6 and higher. If you're using Python 2.5 and want to make use of Fabric's parallel execution features you'll need to install it manually; the recommended route, as usual, is via `pip`. Please see the multiprocessing PyPI page for details.

> **Warning:** Early versions of Python 2.6 (in our testing, 2.6.0 through 2.6.2) ship with a buggy `multiprocessing` module that appears to cause Fabric to hang at the end of sessions involving large numbers of concurrent hosts. If you encounter this problem, either use `env.pool_size` / `-z` to limit the amount of concurrency, or upgrade to Python >=2.6.3.
>
> Python 2.5 is unaffected, as it requires the PyPI version of `multiprocessing`, which is newer than that shipped with Python <2.6.3.

## 4.6 Upgrading from 1.x

Modern Fabric (2+) represents a near-total reimplementation & reorganization of the software. It's been broken in two, cleaned up, made more explicit, and so forth. In some cases, upgrading requires only basic search & replace; in others, more work is needed.

If you read this document carefully, it should guide you in the right direction until you're fully upgraded. If any functionality you're using in Fabric 1 isn't listed here, please file a ticket on Github and we'll update it ASAP.

> **Warning:** As of the 2.0 release line, Fabric 2 is **not** at 100% feature parity with 1.x! Some features have been explicitly dropped, but others simply have not been ported over yet, either due to time constraints or because said features need to be re-examined in a modern context.
>
> Please review the information below, including the *Upgrade specifics* section which contains a very detailed list, before filing bug reports!
>
> Also see *the roadmap* for additional notes about release versioning.

### 4.6.1 Why upgrade?

We'd like to call out, in no particular order, some specific improvements in modern Fabric that might make upgrading worth your time.

**Note:** These are all listed in the rest of the doc too, so if you're already sold, just skip there.

- Python 3 compatibility (specifically, we now support 2.7 and 3.4+);
- Thread-safe - no more requirement on multiprocessing for concurrency;
- API reorganized around `fabric.connection.Connection` objects instead of global module state;
- Command-line parser overhauled to allow for regular GNU/POSIX style flags and options on a per-task basis (no more `fab mytask:weird=custom,arg=format`);
- Task organization is more explicit and flexible / has less 'magic';
- Tasks can declare other tasks to always be run before or after themselves;

- Configuration massively expanded to allow for multiple config files & formats, env vars, per-user/project/module configs, and much more;
- SSH config file loading enabled by default & has been fleshed out re: system/user/runtime file selection;
- Shell command execution API consistent across local and remote method calls - no more differentiation between `local` and `run` (besides where the command runs, of course!);
- Shell commands significantly more flexible re: interactive behavior, simultaneous capture & display (now applies to local subprocesses, not just remote), encoding control, and auto-responding;
- Use of Paramiko's APIs for the SSH layer much more transparent - e.g. `fabric.connection.Connection` allows control over the kwargs given to `SSHClient.connect`;
- Gateway/jump-host functionality offers a `ProxyJump` style 'native' (no proxy-command subprocesses) option, which can be nested infinitely;

### 4.6.2 'Sidegrading' to Invoke

We linked to a note about this above, but to be explicit: modern Fabric is really a few separate libraries, and anything not strictly SSH or network related has been split out into the Invoke project.

This means that if you're in the group of users leveraging Fabric solely for its task execution or `local`, and never used `run`, `put` or similar - **you don't need to use Fabric itself anymore** and can simply **'sidegrade' to Invoke instead**.

You'll still want to read over this document to get a sense of how things have changed, but be aware that you can get away with `pip install invoke` and won't need Fabric, Paramiko, cryptography dependencies, or anything else.

### 4.6.3 Using modern Fabric from within Invoke

We intend to enhance modern Fabric until it encompasses the bulk of Fabric 1's use cases, such that you can use `fab` and fabfiles on their own without caring too much about how it's built on top of Invoke.

However, prior to that point – and very useful on its own for intermediate-to-advanced users – is the fact that modern Fabric is designed with library or direct API use in mind. **It's entirely possible, and in some cases preferable, to use Invoke for your CLI needs and Fabric as a pure API within your Invoke tasks.**

In other words, you can eschew `fab`/fabfiles entirely unless you find yourself strongly needing the conveniences it wraps around ad-hoc sessions, such as `--hosts` and the like.

### 4.6.4 Running both Fabric versions simultaneously

To help with gradual upgrades, modern Fabric may be installed under the name `fabric2` (in addition to being made available "normally" as versions 2.0+ of `fabric`) and can live alongside installations of version 1.x.

Thus, if you have a large codebase and don't want to make the jump to modern versions in one leap, it's possible to have both Fabric 1 (`fabric`, as you presumably had it installed previously) and modern Fabric (as `fabric2`) resident in your Python environment simultaneously.

---

**Note:** We strongly recommend that you eventually migrate all code using Fabric 1, to versions 2 or above, so that you can move back to installing and importing under the `fabric` name. `fabric2` as a distinct package and module is intended to be a stopgap, and there will not be any `fabric3` or above (not least because some of those names are already taken!)

---

For details on how to obtain the `fabric2` version of the package, see *Installing modern Fabric as fabric2*.

### Creating `Connection` and/or `Config` objects from v1 settings

A common tactic when upgrading piecemeal is to generate modern Fabric objects whose contents match the current Fabric 1 environment. Whereas Fabric 1 stores *all* configuration (including the "current host") in a single place – the `env` object – modern Fabric breaks things up into multiple (albeit composed) objects: `Connection` for per-connection parameters, and `Config` for general settings and defaults.

In most cases, you'll only need to generate a `Connection` object using the alternate class constructor `Connection.from_v1`, which should be fed your appropriate local `fabric.api.env` object; see its API docs for details.

A contrived example:

```
from fabric.api import env, run
from fabric2 import Connection

env.host_string = "admin@myserver"
run("whoami") # v1
cxn = Connection.from_v1(env)
cxn.run("whoami") # v2+
```

By default, this constructor calls another API member – `Config.from_v1` – internally on your behalf. Users who need tighter control over modern-style config options may opt to call that classmethod explicitly and hand their modified result into `Connection.from_v1`, which will cause the latter to skip any implicit config creation.

### Mapping of v1 `env` vars to modern API members

The `env` vars and how they map to `Connection` arguments or `Config` values (when fed into the `.from_v1` constructors described above) are listed below.

| v1 `env` var | v2+ usage (prefixed with the class it ends up in) |
|---|---|
| `always_use_pty` | Config: `run.pty`. |
| `forward_agent` | Config: `connect_kwargs.forward_agent`. |
| `gateway` | Config: `gateway`. |
| `host_string` | Connection: `host` kwarg (which can handle host-string like values, including user/port). |
| `key` | **Not supported**: Fabric 1 performed extra processing on this (trying a bunch of key classes to instantiate) before handing it into Paramiko; modern Fabric prefers to just let you handle Paramiko-level parameters directly.<br>If you're filling your Fabric 1 `key` data from a file, we recommend switching to `key_filename` instead, which is supported.<br>If you're loading key data from some other source as a string, you should know what type of key your data is and manually instantiate it instead, then supply it to the `connect_kwargs` parameter. For example:<br>`from io import StringIO  # or 'from StringIO' on Python 2`<br>`from fabric.state import env`<br>`from fabric2 import Connection`<br>`from paramiko import RSAKey`<br>`from somewhere import load_my_key_string`<br>`pkey = RSAKey.from_private_key(StringIO(load_my_key_string()))`<br>`cxn = Connection.from_v1(env, connect_kwargs={"pkey": pkey})` |
| `key_filename` | Config: `connect_kwargs.key_filename`. |
| `no_agent` | Config: `connect_kwargs.allow_agent` (inverted). |
| `password` | Config: `connect_kwargs.password`, as well as `sudo.password` **if and only if** the env's `sudo_password` (see below) is unset. (This mimics how v1 uses this particular setting - in earlier versions there was no `sudo_password` at all.) |
| `port` | Connection: `port` kwarg. Is casted to an integer due to Fabric 1's default being a string value (which is not valid in v2).<br>**Note:** Since v1's `port` is used both for a default *and* to store the current connection state, v2 uses it to fill in the Connection only, and not the Config, on assumption that it will typically be the current connection state. |
| `ssh_config_path` | Config: `ssh_config_path`. |
| `sudo_password` | Config: `sudo.password`. |
| `sudo_prompt` | Config: `sudo.prompt`. |
| `timeout` | Config: `timeouts.connection` (because v1's ambiguously named `timeout` setting was, in fact, for connection timeouts). |
| `use_ssh_config` | Config: `load_ssh_configs`. |
| `user` | Connection: `user` kwarg. |
| `warn_only` | Config: `run.warn` |

### 4.6.5 Upgrade specifics

This is (intended to be) an exhaustive list of *all* Fabric 1.x functionality, as well as new-to-Invoke-or-Fabric-2 functionality not present in 1.x; it specifies whether upgrading is necessary, how to upgrade if so, and tracks features which haven't been implemented in modern versions yet.

Most sections are broken down in table form, as follows:

| Fabric 1 feature or behavior | Status, see below for breakdown | Migration notes, removal rationale, etc |
|---|---|---|

Below are the typical values for the 'status' column, though some of them are a bit loose - make sure to read the notes column in all cases! Also note that things are not ironclad - eg any 'removed' item has some chance of returning if enough users request it or use cases are made that workarounds are insufficient.

- **Ported**: available already, possibly renamed or moved (frequently, moved into the Invoke codebase.)
- **Pending**: would fit, but has not yet been ported, good candidate for a patch. *These entries link to the appropriate Github ticket* - please do not make new ones!
- **Removed**: explicitly *not* ported (no longer fits with vision, had too poor a maintenance-to-value ratio, etc) and unlikely to be reinstated.

Here's a quick local table of contents for navigation purposes:



#### General / conceptual

- Modern Fabric is fully Python 3 compatible; as a cost, Python 2.5 support (a longstanding feature of Fabric 1) has been dropped - in fact, we've dropped support for anything older than Python 2.7.

- The CLI task-oriented workflow remains a primary design goal, but the library use case is no longer a second-class citizen; instead, the library functionality has been designed first, with the CLI/task features built on top of it.
- Additionally, within the CLI use case, version 1 placed too much emphasis on 'lazy' interactive prompts for authentication secrets or even connection parameters, driven in part by a lack of strong configuration mechanisms. Over time it became clear this wasn't worth the tradeoffs of having confusing noninteractive behavior and difficult debugging/testing procedures.

  Modern Fabric takes an arguably cleaner approach (based on functionality added to v1 over time) where users are encouraged to leverage the configuration system and/or serve the user prompts for runtime secrets at the *start* of the process; if the system determines it's missing information partway through, it raises exceptions instead of prompting.
- Invoke's design includes explicit user-facing testing functionality; if you didn't find a way to write tests for your Fabric-using code before, it should be much easier now.
  - We recommend trying to write tests early on; they will help clarify the upgrade process for you & also make the process safer!

### API organization

High level code flow and API member concerns.

| | | |
|---|---|---|
| Import everything via `fabric.api` | Removed | All useful imports are now available at the top level, e.g. `from fabric import Connection`. |
| Configure connection parameters globally (via `env.host_string`, `env.host`, `env.port`, `env.user`) and call global methods which implicitly reference them (`run`/`sudo`/etc) | Removed | The primary API is now properly OOP: instantiate `fabric.connection.Connection` objects and call their methods. These objects encapsulate all connection state (user, host, gateway, etc) and have their own SSH client instances.<br>**See also:**<br>`Connection.from_v1` |
| Emphasis on serialized "host strings" as method of setting user, host, port, etc | Ported/Removed | `fabric.connection.Connection` *can* accept a shorthand "host string"-like argument, but the primary API is now explicit user, host, port, etc keyword arguments.<br>Additionally, many arguments/settings/etc that expected a host string in v1 will now expect a `fabric.connection.Connection` instance instead. |
| Use of "roles" as global named lists of host strings | Ported | This need is now served by `fabric.group.Group` objects (which wrap some number of `fabric.connection.Connection` instances with "do a thing to all members" methods.) Users can create & organize these any way they want.<br>See the line items for `--roles` (*CLI arguments, options and behavior*), `env.roles` (*fabric.env reference*) and `@roles` (*Task functions & decorators*) for the status of those specifics. |

### Task functions & decorators

**Note:** Nearly all task-related functionality is implemented in Invoke; for more details see its execution and namespaces documentation.

| | | |
|---|---|---|
| By default, tasks are loaded from a `fabfile.py` which is sought up towards filesystem root from the user's current working directory | Ported | This behavior is basically identical today, with minor modifications and enhancements (such as tighter control over the load process, and API hooks for implementing custom loader logic - see Loading collections.) |
| "Classic" style implicit task functions lacking a `@task` decorator | Removed | These were on the way out even in v1, and arbitrary task/namespace creation is more explicitly documented now, via Invoke's `Task` and `Collection`. |
| "New" style `@task`-decorated, module-level task functions | Ported | Largely the same, though now with superpowers - `@task` can still be used without any parentheses, but where v1 only had a single `task_class` argument, the new version (largely based on Invoke's) has a number of namespace and parser hints, as well as execution related options (such as those formerly served by `@hosts` and friends). |
| Arbitrary task function arguments (i.e. `def mytask(any, thing, at, all)`) | Ported | This gets its own line item because: tasks must now take a `Context` (vanilla Invoke) or `fabric.connection.Connection` (Fabric) object as their first positional argument. The rest of the function signature is, as before, totally up to the user & will get automatically turned into CLI flags.<br>This sacrifices a small bit of the "quick DSL" of v1 in exchange for a cleaner, easier to understand/debug, and more user-overrideable API structure.<br>As a side effect, it lessens the distinction between "module of functions" and "class of methods"; users can more easily start with the former and migrate to the latter when their needs grow/change. |
| Implicit task tree generation via import-crawling | Ported/Removed | Namespace construction is now more explicit; for example, imported modules in your `fabfile.py` are no longer auto-scanned and auto-added to the task tree.<br>However, the root `fabfile.py` *is* automatically loaded (using `Collection.from_module`), preserving the simple/common case. See Constructing namespaces for details.<br>We may reinstate (in an opt-in fashion) imported module scanning later, since the use of explicit namespace objects still allows users control over the tree that results. |
| `@hosts` for determining the default host or list of hosts a given task uses | Ported | Reinstated as the `hosts` parameter of `@task`. Further, it can now handle dicts of `fabric.connection.Connection` kwargs in addition to simple host strings. |
| `@roles` for determining the default list of group-of-host targets a given task uses | Pending | See *API organization* for details on the overall 'roles' concept. When it returns, this will probably follow `@hosts` and become some `@task` argument. |
| `@serial`/`@parallel`/`@runs_once` | Ported/Pending | Parallel execution is currently offered at the API level via `fabric.group.Group` subclasses such as `fabric.group.ThreadingGroup`; however, designating entire sessions and/or tasks to run in parallel (or to exempt from parallelism) has not been solved yet. The problem needs solving at a higher level than just SSH targets, so this links to an Invoke-level ticket. |
| `execute` for calling named tasks from other tasks while honoring decorators and other execution mechanics (as opposed to calling them simply as functions) | Pending | This is one of the top "missing features" from the rewrite; link is to Invoke's tracker. |
| `Task` class for programmatic creation of tasks (as opposed to using some function object and the `@task` decorator) | Ported | While not sharing many implementation details with v1, modern Fabric (via Invoke) has a publicly exposed `Task` class, which alongside `Collection` al- |

### CLI arguments, options and behavior

| | | |
|---|---|---|
| Exposure of task arguments as custom colon/comma delimited CLI arguments, e.g. `fab mytask:posarg,kwarg=val` | Removed | CLI arguments are now proper GNU/POSIX-style long and short flags, including globbing shortflags together, space or equals signs to attach values, optional values, and much more. See Invoking tasks. |
| Task definition names are mirrored directly on the command-line, e.g for task `def journald_logs()`, command line argument is `fab journald_logs` | Removed | Tasks names now get converted from underscores to hyphens. Eg. task `def journald_logs()` now evaluates to `fab journald-logs` on the commandline. |
| Ability to invoke multiple tasks in a single command line, e.g. `fab task1 task2` | Ported | Works great! |
| `python -m fabric` as stand-in for `fab` | Ported | Ported in 2.2. |
| `-a`/`--no_agent` for disabling automatic SSH agent key selection | Removed | To disable use of an agent permanently, set config value `connect_kwargs.allow_agent` to `False`; to disable temporarily, unset the `SSH_AUTH_SOCK` env var. |
| `-A`/`--forward-agent` for enabling agent forwarding to the remote end | Removed | The config and kwarg versions of this are ported, but there is currently no CLI flag. Usual "you can set the config value at runtime with a shell env variable" clause is in effect, so this *may* not get ported, depending. |
| `--abort-on-prompts` to turn interactive prompts into exceptions (helps avoid 'hanging' sessions) | Removed | See the notes about interactive prompts going away in *General / conceptual*. Without mid-session prompts, there's no need for this option. |
| `-c`/`--config` for specifying an alternate config file path | Ported | `--config` lives on, but the short flag is now `-f` (`-c` now determines which collection module name is sought by the task loader.) |
| `--colorize-errors` (and `env.colorize_errors`) to enable ANSI coloring of error output | Pending | Very little color work has been done yet and this is one of the potentially missing pieces. We're unsure how often this was used in v1 so it's possible it won't show up again, but generally, we like using color as an additional output vector, so... |
| `-d`/`--display` for showing info on a given command | Ported | This is now the more standard `-h`/`--help`, and can be given in either "direction": `fab -h mytask` or `fab mytask -h`. |
| `-D`/`--disable-known-hosts` to turn off Paramiko's automatic loading of user-level `known_hosts` files | Pending | Not ported yet, probably will be. |
| `-e`/`--eagerly-disconnect` (and `env.eagerly_disconnect`) which tells the execution system to disconnect from hosts as soon as a task is done running | Ported/Pending | There's no explicit connection cache anymore, so eager disconnection should be less necessary. However, investigation and potential feature toggles are still pending. |
| `-f`/`--fabfile` to select alternate fabfile location | Ported | This is now split up into `-c`/`--collection` and `-r`/`--search-root`; see Loading collections. |

Continued on next page

Table 4.1 – continued from previous page

| | | |
|---|---|---|
| `-g`/`--gateway` (and `env.gateway`) for selecting a global SSH gateway host string | Pending | One can set the global `gateway` config option via an environment variable, which at a glance would remove the need for a dedicated CLI option. However, this approach only allows setting string values, which in turn only get used for `ProxyCommand` style gatewaying, so it *doesn't* replace v1's `--gateway` (which took a host string and turned it into a `ProxyJump` style gateway). Thus, if enough users notice the lack, we'll consider a feature-add that largely mimics the v1 behavior: string becomes first argument to `fabric.connection.Connection` and that resulting object is then set as `gateway`. |
| `--gss-auth`/`--gss-deleg`/`--gss-kex` | Removed | These didn't seem used enough to be worth porting over, especially since they fall under the usual umbrella of "Paramiko-level connect passthrough" covered by the `connect_kwargs` config option. (Which, if necessary, can be set at runtime via shell environment variables, like any other config value.) |
| `--hide`/`--show` for tweaking output display globally | Removed | This is configurable via the config system and env vars. |
| `-H`/`--hosts` | Ported | Works basically the same as before - if given, is shorthand for executing any given tasks once per host. |
| `-i` for SSH key filename selection | Ported | Works same as v1, including ability to give multiple times to build a list of keys to try. |
| `-I`/`--initial-password-prompt` for requesting an initial pre-execution password prompt | Ported | It's now `--prompt-for-login-password`, –prompt-for-sudo-password or `--prompt-for-passphrase`, depending on whether you were using the former to fill in passwords or key passphrases (or both.) |
| `--initial-sudo-password-prompt` for requesting an initial pre-execution sudo password prompt | Ported | This is now `--prompt-for-sudo-password`. Still a bit of a mouthful but still 4 characters shorter! |
| `-k`/`--no-keys` which prevents Paramiko's automatic loading of key files such as `~/.ssh/id_rsa` | Removed | Use environment variables to set the `connect_kwargs.look_for_keys` config value to `False`. |
| `--keepalive` for setting network keepalive | Pending | Not ported yet. |
| `-l`/`--list` for listing tasks, plus `-F`/`--list-format` for tweaking list display format | Ported | Now with bonus JSON list-format! Which incidentally replaces `-F short`/`--shortlist`. |
| `--linewise` for buffering output line by line instead of roughly byte by byte | Removed | This doesn't really fit with the way modern command execution code views the world, so it's gone. |
| `-n`/`--connection-attempts` controlling multiple connect retries | Pending | Not ported yet. |
| `--no-pty` to disable automatic PTY allocation in `run`, etc | Ported | Is now `-p`/`--pty` as the default behavior was switched around. |
| `--password`/`--sudo-password` for specifying login/sudo password values | Removed | This is typically not very secure to begin with, and there are now many other avenues for setting the related configuration values, so they're gone at least for now. |
| `-P`/`--parallel` for activating global parallelism | Pending | See the notes around `@parallel` in *Task functions & decorators*. |

Continued on next page

Table 4.1 – continued from previous page

| | | |
|---|---|---|
| `--port` to set default SSH port | Removed | Our gut says this is best left up to the configuration system's env var layer, or use of the `port` kwarg on `fabric.connection.Connection`; however it may find its way back. |
| `-r`/`--reject-unknown-hosts` to modify Paramiko known host behavior | Pending | Not ported yet. |
| `-R`/`--roles` for global list-of-hosts target selection | Pending | As noted under *API organization*, role lists are only partially applicable to the new API and we're still feeling out whether/how they would work at a global or CLI level. |
| `--set key=value` for setting `fabric.state.env` vars at runtime | Removed | This is largely obviated by the new support for shell environment variables (just do `INVOKE_KEY=value fab mytask` or similar), though it's remotely possible a CLI flag method of setting config values will reappear later. |
| `-s`/`--shell` to override default shell path | Removed | Use the configuration system for this. |
| `--shortlist` for short/computer-friendly list output | Ported | See `--list`/`--list-format` - there's now a JSON format instead. No point reinventing the wheel. |
| `--skip-bad-hosts` (and `env.skip_bad_hosts`) to bypass problematic hosts | Pending | Not ported yet. |
| `--skip-unknown-tasks` and `env.skip_unknown_tasks` for silently skipping past bogus task names on CLI invocation | Removed | This felt mostly like bloat to us and could require non-trivial parser changes to reimplement, so it's out for now. |
| `--ssh-config-path` and `env.ssh_config_path` for selecting an SSH config file | Ported | This is now `-S`/`--ssh-config`. |
| `--system-known-hosts` to trigger loading systemwide `known_hosts` files | Pending/Removed | This isn't super likely to come back as its own CLI flag but it may well return as a configuration value. |
| `-t`/`--timeout` controlling connection timeout | Ported | This is now part of the direct passthrough to Paramiko-level connection parameters, the `connect_kwargs` config value. |
| `-T`/`--command-timeout` | Pending | See notes in *Shell command execution (local/run/sudo)* around the `timeout` kwarg. |
| `-u`/`--user` to set global default username | Removed | Most of the time, configuration (env vars for true runtime, or eg user/project level config files as appropriate) should be used for this, but it may return. |
| `-w`/`--warn-only` to toggle warn-vs-abort behavior | Ported | Ported as-is, no changes. |
| `-x`/`--exclude-hosts` (and `env.exclude_hosts`) for excluding otherwise selected targets | Pending | Not ported yet, is pending an in depth rework of global (vs hand-instantiated) connection/group selection. |
| `-z`/`--pool-size` for setting parallel-mode job queue pool size | Removed | There's no job queue anymore, or at least at present. Whatever replaces it (besides the already-implemented threading model) is likely to look pretty different. |

### Shell command execution (`local`/`run`/`sudo`)

### General

Behaviors shared across either `run`/`sudo`, or all of `run`/`sudo`/`local`. Subsequent sections go into per-function differences.

| | | |
|---|---|---|
| `local` and `run`/`sudo` have wildly differing APIs and implementations | Removed | All command execution is now unified; all three functions (now methods on `fabric.connection.Connection`, though `local` is also available as `invoke.run` for standalone use) have the same underlying protocol and logic (the `Runner` class hierarchy), with only low-level details like process creation and pipe consumption differing.<br>For example, in v1 `local` required you to choose between displaying and capturing subprocess output; modern `local` is like `run` and does both at the same time. |
| Prompt auto-response, via `env.prompts` and/or `sudo`'s internals | Ported | The `env.prompts` functionality has been significantly fleshed out, into a framework of Watchers which operate on any (local or remote!) running command's input and output streams.<br>In addition, `sudo` has been rewritten to use that framework; while still useful enough to offer an implementation in core, it no longer does anything users cannot do themselves using public APIs. |
| `fabric.context_managers.cd`/`lcd` (and `prefix`) allow scoped mutation of executed comments | Ported/Pending | These are now methods on `Context` (`Context.cd`, `Context.prefix`) but need work in its subclass `fabric.connection.Connection` (quite possibly including recreating `lcd`) so that local vs remote state are separated. |
| `fabric.context_managers.shell_env` and its specific expression `path` (plus `env.shell_env`, `env.path` and `env.path_behavior`), for modifying remote environment variables (locally, one would just modify `os.environ`.) | Ported | The context managers were the only way to set environment variables at any scope; in modern Fabric, subprocess shell environment is controllable per-call (directly in `fabric.connection.Connection.run` and siblings via an `env` kwarg) *and* across multiple calls (by manipulating the configuration system, statically or at runtime.) |
| Controlling subprocess output & other activity display text by manipulating `fabric.state.output` (directly or via `fabric.context_managers.hide`, `show` or `quiet` as well as the `quiet` kwarg to `run`/`sudo`; plus `utils.puts`/`fastprint`) | Ported/Pending | The core concept of "output levels" is gone, likely to be replaced in the near term by a logging module (stdlib or other) which output levels poorly reimplemented.<br>Command execution methods like `run` retain a `hide` kwarg controlling which subprocess streams are copied to your terminal, and an `echo` kwarg controlling whether commands are printed before execution. All of these also honor the configuration system. |
| `timeout` kwarg and the `CommandTimeout` exception raised when said command-runtime timeout was violated | Pending | Command timeouts have not been ported yet, but will likely be added (at the Invoke layer) in future. |
| `pty` kwarg and `env.always_use_pty`, controlling whether commands run in a pseudo-terminal or are invoked directly | Ported | This has been thoroughly ported (and its behavior often improved) including preservation of the `pty` kwarg and updating the config value to be simply `run.pty`. However, a major change is that pty allocation is now `False` by default instead of `True`.<br>Fabric 0.x and 1.x already changed this value around; during Fabric 1's long lifetime it became clear that neither default works for all or even most users, so we opted to return the default to `False` as it's cleaner and less wasteful. |
| `combine_stderr` (kwarg and `env.combine_stderr`) controlling whether Paramiko weaves remote stdout and stderr into the stdout stream | Removed | This wasn't terrifically useful, and often caused conceptual problems in tandem with `pty` (as pseudo-terminals by their nature always combine the two streams).<br>We recommend users who really need both streams to be merged, either use shell redirection in their command, or set `pty=True`. |

#### run

| | | |
|---|---|---|
| `shell` / `env.use_shell` designating whether or not to wrap commands within an explicit call to e.g. `/bin/sh -c "real command"`; plus their attendant options like `shell_escape` | Removed | Non-`sudo` remote execution never truly required an explicit shell wrapper: the remote SSH daemon hands your command string off to the connecting user's login shell in almost all cases. Since wrapping is otherwise extremely error-prone and requires frustrating escaping rules, we dropped it for this use case.<br>See the matching line items for `local` and `sudo` as their situations differ. (For now, because they all share the same underpinnings, `fabric.connection.Connection.run` does accept a `shell` kwarg - it just doesn't do anything with it.) |

#### sudo

Unless otherwise noted, all common `run``+``sudo` args/functionality (e.g. `pty`, `warn_only` etc) are covered above in the section on `run`; the below are `sudo` specific.

| | | |
|---|---|---|
| `shell` / `env.use_shell` designating whether or not to wrap commands within an explicit call to e.g. `/bin/sh -c "real command"` | Pending/Removed | See the note above under `run` for details on shell wrapping as a general strategy; unfortunately for `sudo`, some sort of manual wrapping is still necessary for non-trivial commands (i.e. anything using actual shell syntax as opposed to a single program's argv) due to how the command string is handed off to the `sudo` program.<br>We hope to upgrade `sudo` soon so it can perform a common-best-case, no-escaping-required shell wrapping on your behalf; see the 'Pending' link. |
| `user` argument (and `env.sudo_user`) allowing invocation via `sudo -u <user>` (instead of defaulting to root) | Ported | This is still here, and still called `user`. |
| `group` argument controlling the effective group of the sudo'd command | Pending | This has not been ported yet. |

#### local

See the 'general' notes at top of this section for most details about the new `local`. A few specific extras are below.

| | | |
|---|---|---|
| `shell` kwarg designating which shell to ask `subprocess.Popen` to use | Ported | Basically the same as in v1, though there are now situations where `os.execve` (or similar) is used instead of `subprocess.Popen`. Behavior is much the same: no shell wrapping (as in legacy `run`), just informing the operating system what actual program to run. |

### Utilities

| | | |
|---|---|---|
| Error handling via `abort` and `warn` | Ported | The old functionality leaned too far in the "everything is a DSL" direction & didn't offer enough value to offset how it gets in the way of experienced Pythonistas.<br>These functions have been removed in favor of "just raise an exception" (with one useful option being Invoke's `Exit`) as exception handling feels more Pythonic than thin wrappers around `sys.exit` or having to `except SystemExit:` and hope it was a `SystemExit` your own code raised! |
| ANSI color helpers in `fabric.colors` allowed users to easily print ANSI colored text without a standalone library | Removed | There seemed no point to poorly replicating one of the many fine terminal-massaging libraries out there (such as those listed in the description of #101) in the rewrite, so we didn't.<br>That said, it seems highly plausible we'll end up vendoring such a library in the future to offer internal color support, at which point "baked-in" color helpers would again be within easy reach. |
| `with char_buffered` context manager for forcing a local stream to be character buffered | Ported | This is now `character_buffered`. |
| `docs.unwrap_tasks` for extracting docstrings from wrapped task functions | Ported | v1 required using a Fabric-specific 'unwrap_tasks' helper function somewhere in your Sphinx build pipeline; now you can instead just enable the new invocations.autodoc Sphinx mini-plugin in your extensions list; see link for details. |
| `network.normalize`, `denormalize` and `parse_host_string`, ostensibly internals but sometimes exposed to users for dealing with host strings | Removed | As with other host-string-related tools, these are gone and serve no purpose. `fabric.connection.Connection` is now the primary API focus and has individual attributes for all "host string" components. |
| `utils.indent` for indenting/wrapping text (uncommonly used) | Pending | Not ported yet; ideally we'll just vendor a third party lib in Invoke. |
| `reboot` for rebooting and reconnecting to a remote system | Removed | No equivalent has been written for modern Fabric; now that the connection/client objects are made explicit, one can simply instantiate a new object with the same parameters (potentially with sufficient timeout parameters to get past the reboot, if one doesn't want to manually call something like `time.sleep`.)<br>There is a small chance it will return if there appears to be enough need; if so, it's likely to be a more generic reconnection related `fabric.connection.Connection` method, where the user is responsible for issuing the restart shell command via `sudo` themselves. |
| `require` for ensuring certain key(s) in `env` have values set, optionally by noting they can be `provided_by=` a list of setup tasks | Removed | This has not been ported, in part because the maintainers never used it themselves, and is unlikely to be directly reimplemented. However, its core use case of "require certain data to be available to run a given task" may return within the upcoming dependency framework. |
| `prompt` for prompting the user & storing the entered data (optionally with validation) directly into `env` | Removed | Like `require`, this seemed like a less-used feature (especially compared to its sibling `confirm`) and was not ported. If it returns it's likely to be via `invocations`, which is where `confirm` ended up. |

### Networking

| | | |
|---|---|---|
| `env.gateway` for setting an SSH jump gateway | Ported | This is now the `gateway` kwarg to `fabric.connection.Connection`, and – for the newly supported `ProxyJump` style gateways, which can be nested indefinitely! – should be another `fabric.connection.Connection` object instead of a host string.<br>(You may specify a runtime, non-SSH-config-driven `ProxyCommand`-style string as the `gateway` kwarg instead, which will act just like a regular `ProxyCommand`.) |
| `ssh_config`-driven `ProxyCommand` support | Ported | This continues to work as it did in v1. |
| `with remote_tunnel(...):` port forwarding | Ported | This is now `fabric.connection.Connection.forward_local`, since it's used to *forward* a *local* port to the remote end. (Newly added is the logical inverse, `fabric.connection.Connection.forward_remote`.) |
| `NetworkError` raised on some network related errors | Removed | In v1 this was simply a (partially implemented) stepping-back from the original "just sys.exit on any error!" behavior. Modern Fabric is significantly more exception-friendly; situations that would raise `NetworkError` in v1 now simply become the real underlying exceptions, typically from Paramiko or the stdlib. |
| `env.keepalive` for setting network keepalive value | Pending | Not ported yet. |
| `env.connection_attempts` for setting connection retries | Pending | Not ported yet. |
| `env.timeout` for controlling connection timeout | Ported | This is now controllable both via the configuration system and a direct kwarg on `fabric.connection.Connection`. |

### Authentication

**Note:** Some `env` keys from v1 were simply passthroughs to Paramiko's `SSHClient.connect` method. Modern Fabric gives you explicit control over the arguments it passes to that method, via the `connect_kwargs` configuration subtree, and the below table will frequently refer you to that approach.

| | | |
|---|---|---|
| `env.key_filename` | Ported | Use `connect_kwargs`. |
| `env.password` | Ported | Use `connect_kwargs`.<br>Also note that this used to perform double duty as connection *and* sudo password; the latter is now found in the `sudo.password` setting. |
| `env.gss_(auth\|deleg\|kex)` | Ported | Use `connect_kwargs`. |
| `env.key`, a string or file object holding private key data, whose specific type is autodetermined and instantiated for use as the `pkey` connect kwarg | Removed | This has been dropped as unnecessary (& bug-prone) obfuscation of Paramiko-level APIs; users should already know which type of key they're dealing with and instantiate a `PKey` subclass themselves, placing the result in `connect_kwargs.pkey`. |
| `env.no_agent`, which is a renaming/inversion of Paramiko's `allow_agent` connect kwarg | Ported | Users who were setting this to `True` should now simply set `connect_kwargs.allow_agent` to `False` instead. |
| `env.no_keys`, similar to `no_agent`, just an inversion of the `look_for_keys` connect kwarg | Ported | Use `connect_kwargs.look_for_keys` instead (setting it to `False` to disable Paramiko's default key-finding behavior.) |
| `env.passwords` (and `env.sudo_passwords`) stores connection/sudo passwords in a dict keyed by host strings | Ported/Pending | Each `fabric.connection.Connection` object may be configured with its own `connect_kwargs` given at instantiation time, allowing for per-host password configuration already.<br>However, we expect users may want a simpler way to set configuration values that are turned into implicit `fabric.connection.Connection` objects automatically; such a feature is still pending. |
| Configuring `IdentityFile` in one's `ssh_config` | Ported | Still honored, along with a bunch of newly honored `ssh_config` settings; see Loading and using ssh_config files. |

### File transfer

The below feature breakdown applies to the `put` and/or `get` "operation" functions from v1.

| | | |
|---|---|---|
| Transferring individual files owned by the local and remote user | Ported | Basic file transfer in either direction works and is offered as `fabric.connection.Connection.get`/`fabric.connection.Connection.put` (though the code is split out into a separate-responsibility class, `fabric.transfer.Transfer`.)<br>The signature of these methods has been cleaned up compared to v1, though their positional-argument essence (`get(remote, local)` and `put(local, remote)` remains the same. |
| Omit the 'destination' argument for implicit 'relative to local context' behavior (e.g. `put("local.txt")` implicitly uploading to remote `$HOME/local.txt`.) | Ported | You should probably still be explicit, because this is Python. |
| Use either file paths *or* file-like objects on either side of the transfer operation (e.g. uploading a `StringIO` instead of an on-disk file) | Ported | This was a useful enough and simple enough trick to keep around. |
| Preservation of source file mode at destination (e.g. ensuring an executable bit that would otherwise be dropped by the destination's umask, is re-added.) | Ported | Not only was this ported, but it is now the default behavior. It may be disabled via kwarg if desired. |
| Bundled `sudo` operations as part of file transfer | Removed | This was one of the absolute buggiest parts of v1 and never truly did anything users could not do themselves with a followup call to `sudo`, so we opted not to port it. Should enough users pine for its loss, we *may* reconsider, but if we do it will be with a serious eye towards simplification and/or an approach not involving intermediate files. |
| Recursive multi-file transfer (e.g. `put(a_directory)` uploads entire directory and all its contents) | Removed | This was *another* one of the buggiest parts of v1, and over time it became clear that its maintenance burden far outweighed the fact that it was poorly reinventing `rsync` and/or the use of archival file tools like ye olde `tar``+``gzip`.<br>For one potential workaround, see the `rsync` function in patchwork. |
| Remote file path tilde expansion | Removed | This behavior is ultimately unnecessary (one can simply leave the tilde off for the same result) and had a few pernicious bugs of its own, so it's gone. |
| Naming downloaded files after some aspect of the remote destination, to avoid overwriting during multi-server actions | Pending | This falls under the `Group` family, which still needs some work in this regard. |

### Configuration

In general, configuration has been massively improved over the old `fabricrc` files; most config logic comes from Invoke's configuration system, which offers a full-fledged configuration hierarchy (in-code config, multiple config file locations, environment variables, CLI flags, and more) and multiple file formats. Nearly all configuration avenues in Fabric 1 become, in modern Fabric, manipulation of whatever part of the config hierarchy is most appropriate for your needs.

Modern versions of Fabric only make minor modifications to (or parameterizations of) Invoke's setup; see our locally-

specific config doc page for details.

**Note:** Make sure to look elsewhere in this document for details on any given v1 `env` setting, as many have moved outside the configuration system into object or method keyword arguments.

| | | |
|---|---|---|
| Modifying `fabric.(api.)env` directly | Ported | To effect truly global-scale config changes, use config files, task-collection-level config data, or the invoking shell's environment variables. |
| Making locally scoped `fabric.env` changes via `with settings(...):` or its decorator equivalent, `@with_settings` | Ported/Pending | Most of the use cases surrounding `settings` are now served by the fact that `fabric.connection.Connection` objects keep per-host/connection state - the pattern of switching the implicit global context around was a design antipattern which is now gone.<br>The remaining such use cases have been turned into context-manager methods of `fabric.connection.Connection` (or its parent class), or have such methods pending. |
| SSH config file loading (off by default, limited to `~/.ssh/config` only unless configured to a different, single path) | Ported | Much improved: SSH config file loading is **on** by default (which can be changed), multiple sources are loaded and merged just like OpenSSH, and more besides; see Loading and using ssh_config files.<br>In addition, we've added support for some `ssh_config` directives which were ignored by v1, such as `ConnectTimeout` and `ProxyCommand`, and going forwards we intend to support as much of `ssh_config` as is reasonably possible. |

### contrib

The old `contrib` module represented "best practice" functions that did not, themselves, require core support from the rest of Fabric but were built using the same primitives available to users.

In modern Fabric, that responsibility has been removed from the core library into other standalone libraries which have their own identity & release process, typically either invocations (local-oriented code that does not use SSH) or patchwork (primarily remote-oriented code, though anything not explicitly dealing with both ends of the connection will work just as well locally.)

Those libraries are still a work in progress, not least because we still need to identify the best way to bridge the gap between them (as many operations are not intrinsically local-or-remote but can work on either end.)

Since they are by definition built on the core APIs available to all users, they currently get less development focus; users can always implement their own versions without sacrificing much (something less true for the core libraries.) We expect to put more work into curating these collections once the core APIs have settled down.

Details about what happened to each individual chunk of `fabric.contrib` are in the below table:

| | | |
|---|---|---|
| `console.confirm` for easy bool-returning confirmation prompts | Ported | Moved to `invocations.console.confirm`, with minor signature tweaks. |
| `django.*`, supporting integration with a local Django project re: importing and using Django models and other code | Removed | We aren't even sure if this is useful a decade after it was written, given how much Django has surely changed since then. If you're reading and are sad that this is gone, let us know! |
| `files.*` (e.g. `exists`, `append`, `contains` etc) for interrogating and modifying remote files | Ported/Pending | Many of the more useful functions in this file have been ported to `patchwork.files` but are still in an essentially alpha state.<br>Others, such as `is_link`, `comment`/`uncomment`, etc have not been ported yet. If they are, the are likely to end up in the same place. |
| `project.rsync_project` for rsyncing the entire host project remotely | Ported | Now `patchwork.transfers.rsync`, with some modifications. |
| `project.rsync_project` for uploading host project via archive file and scp | Removed | This did not seem worth porting; the overall pattern of "copy my local bits remotely" is already arguably an antipattern (vs repeatable deploys of artifacts, or at least remote checkout of a VCS tag) and if one is going down that road anyways, rsync is a much smarter choice. |

### `fabric.env` reference

Many/most of the members in v1's `fabric.env` are covered in the above per-topic sections; any that are *not* covered elsewhere, live here. All are explicitly noted as `env.<name>` for ease of searching in your browser or viewer.

A small handful of env vars were never publicly documented & were thus implicitly private; those are not represented here.

| | | |
|---|---|---|
| `env.abort_exception` for setting which exception is used to abort | Removed | Aborting as a concept is gone, just raise whatever exception seems most reasonable to surface to an end user, or use `Exit`. See also *Utilities*. |
| `env.all_hosts` and `env.tasks` listing execution targets | Ported/Pending | Fabric's `Executor` subclass stores references to all CLI parsing results (including the value of `--hosts`, the tasks requested and their args, etc) and the intent is for users to have access to that information.<br>However, the details for that API (e.g. exposing the executor via a task's `Context`/`fabric.connection.Connection`) are still in flux. |
| `env.command` noting currently executing task name (in hindsight, quite the misnomer...) | Ported/Pending | See the notes for `env.all_hosts` above - same applies here re: user visibility into CLI parsing results. |
| `env.command_prefixes` for visibility into (arguably also mutation of) the shell command prefixes to be applied to `run`/`sudo` | Ported | This is now `command_prefixes`. |
| `env.cwd` noting current intended working directory | Ported | This is now `command_cwds` (a list, not a single string, to more properly model the intended contextmanager-driven use case.)<br>Note that remote-vs-local context for this data isn't yet set up; see the notes about `with cd` under *Shell command execution (local/run/sudo)*. |
| `env.dedupe_hosts` controlling whether duplicate hosts in merged host lists get deduplicated or not | Pending | Not ported yet, will probably get tackled as part of roles/host lists overhaul. |
| `env.echo_stdin` (undocumented) for turning off the default echoing of standard input | Ported | Is now a config option under the `run` tree, with much the same behavior. |
| `env.local_user` for read-only access to the discovered local username | Removed | We're not entirely sure why v1 felt this was worth caching in the config; if you need this info, just import and call `fabric.util.get_local_user`. |
| `env.output_prefix` determining whether or not line-by-line host-string prefixes are displayed | Pending | Differentiating parallel stdout/err is still a work in progress; we may end up reusing line-by-line logging and prefixing (ideally via actual logging) or we may try for something cleaner such as streaming to per-connection log files. |
| `env.prompts` controlling prompt auto-response | Ported | Prompt auto-response is now publicly implemented as the `StreamWatcher` and `Responder` class hierarchy, instances of which can be handed to `run` via kwarg or stored globally in the config as `run.watchers`. |
| `env.real_fabfile` storing read-only fabfile path which was loaded by the CLI machinery | Ported | The loaded task `Collection` is stored on both the top level `Program` object as well as the `Executor` which calls tasks; and `Collection` has a `loaded_from` attribute with this information. |
| `env.remote_interrupt` controlling how interrupts (i.e. a local `KeyboardInterrupt` are caught, forwarded or other | Ported/Removed | Invoke's interrupt capture behavior is currently "always just send the interrupt character to the subprocess and continue", allowing subprocesses to handle `^C` however they need to, which is an improvement over Fabric 1 and roughly equivalent to setting `env.remote_interrupt = True`.<br>Allowing users to change this behavior via config is not yet implemented, and may not be, depending on whether anybody needs it - it was added as an option in v1 for backwards compat reasons.<br>It is also technically possible to change interrupt behavior by subclassing and overriding `invoke.runners.Runner.send_interrupt`. |
| `env.roles`, `env.roledefs` and `env.` | Pending | As noted in *API organization*, roles as a concept were |

### 4.6.6 Example upgrade process

This section goes over upgrading a small but nontrivial Fabric 1 fabfile to work with modern Fabric. It's not meant to be exhaustive, merely illustrative; for a full list of how to upgrade individual features or concepts, see *Upgrade specifics*.

#### Sample original fabfile

Here's a (slightly modified to concur with 'modern' Fabric 1 best practices) copy of Fabric 1's final tutorial snippet, which we will use as our test case for upgrading:

```
from fabric.api import abort, env, local, run, settings, task
from fabric.contrib.console import confirm

env.hosts = ["my-server"]

@task
def test():
    with settings(warn_only=True):
        result = local("./manage.py test my_app", capture=True)
    if result.failed and not confirm("Tests failed. Continue anyway?"):
        abort("Aborting at user request.")

@task
def commit():
    local("git add -p && git commit")

@task
def push():
    local("git push")

@task
def prepare_deploy():
    test()
    commit()
    push()

@task
def deploy():
    code_dir = "/srv/django/myproject"
    with settings(warn_only=True):
        if run("test -d {}".format(code_dir)).failed:
            cmd = "git clone user@vcshost:/path/to/repo/.git {}"
            run(cmd.format(code_dir))
    with cd(code_dir):
        run("git pull")
        run("touch app.wsgi")
```

We'll port this directly, meaning the result will still be `fabfile.py`, though we'd like to note that writing your code in a more library-oriented fashion - even just as functions not wrapped in `@task` - can make testing and reusing code easier.

#### Imports

In modern Fabric, we don't need to import nearly as many functions, due to the emphasis on object methods instead of global functions. We only need the following:

- `Exit`, a friendlier way of requesting a `sys.exit`;
- `@task`, as before, but coming from Invoke as it's not SSH-specific;
- `confirm`, which now comes from the Invocations library (also not SSH-specific; though Invocations is one of the descendants of `fabric.contrib`, which no longer exists);

```
from fabric import task
from invoke import Exit
from invocations.console import confirm
```

### Host list

The idea of a predefined *global* host list is gone; there is currently no direct replacement. In general, users can set up their own execution context, creating explicit `fabric.connection.Connection` and/or `fabric.group.Group` objects as needed; core Fabric is in the process of building convenience helpers on top of this, but "create your own Connections" will always be there as a backstop.

Speaking of convenience helpers: most of the functionality of `fab --hosts` and `@hosts` has been ported over – the former directly (see `--hosts`), the latter as a `@task` keyword argument. Thus, for now our example will be turning the global `env.hosts` into a lightweight module-level variable declaration, intended for use in the subsequent calls to `@task`:

```
my_hosts = ["my-server"]
```

**Note:** This is an area under active development, so feedback is welcomed.

### Test task

The first task in the fabfile uses a good spread of the API. We'll outline the changes here (though again, all details are in *Upgrade specifics*):

- Declaring a function as a task is nearly the same as before: use a `@task` decorator (which, in modern Fabric, can take more optional keyword arguments than its predecessor, including some which replace some of v1's decorators).
- `@task`-wrapped functions must now take an explicit initial context argument, whose value will be a `fabric.connection.Connection` object at runtime.
- The use of `with settings(warn_only=True)` can be replaced by a simple kwarg to the `local` call.
- That `local` call is now a method call on the `fabric.connection.Connection`, `fabric.connection.Connection.local`.
- `capture` is no longer a useful argument; we can now capture and display at the same time, locally or remotely. If you don't actually *want* a local subprocess to mirror its stdout/err while it runs, you can simply say `hide=True` (or `hide="stdout"` or etc.)
- Result objects are pretty similar between versions; modern Fabric's results no longer pretend to "be" strings, but instead act more like booleans, acting truthy if the command exited cleanly, and falsey otherwise. In terms of attributes exhibited, most of the same info is available, and more besides.
- `abort` is gone; you should use whatever exceptions you feel are appropriate, or `Exit` for a `sys.exit` equivalent. (Or just call `sys.exit` if you want a no-questions-asked immediate exit that even our CLI machinery won't touch.)

The result:

```
@task
def test(c):
    result = c.local("./manage.py test my_app", warn=True)
    if not result and not confirm("Tests failed. Continue anyway?"):
        raise Exit("Aborting at user request.")
```

### Other simple tasks

The next two tasks are simple one-liners, and you've already seen what replaced the global `local` function:

```
@task
def commit(c):
    c.local("git add -p && git commit")

@task
def push(c):
    c.local("git push")
```

### Calling tasks from other tasks

This is another area that is in flux at the Invoke level, but for now, we can simply call the other tasks as functions, just as was done in v1. The main difference is that we want to pass along our context object to preserve the configuration context (such as loaded config files or CLI flags):

```
@task
def prepare_deploy(c):
    test(c)
    commit(c)
    push(c)
```

### Actual remote steps

Note that up to this point, nothing truly Fabric-related has been in play - `fabric.connection.Connection.local` is just a rebinding of `Context.run`, Invoke's local subprocess execution method. Now we get to the actual deploy step, which invokes `fabric.connection.Connection.run` instead, executing remotely (on whichever host the `fabric.connection.Connection` has been bound to).

`with cd` is not fully implemented for the remote side of things, but we expect it will be soon. For now we fall back to command chaining with `&&`. And, notably, now that we care about selecting host targets, we refer to our earlier definition of a default host list – `my_hosts` – when declaring the default host list for this task.

```
@task(hosts=my_hosts)
def deploy(c):
    code_dir = "/srv/django/myproject"
    if not c.run("test -d {}".format(code_dir), warn=True):
        cmd = "git clone user@vcshost:/path/to/repo/.git {}"
        c.run(cmd.format(code_dir))
    c.run("cd {} && git pull".format(code_dir))
    c.run("cd {} && touch app.wsgi".format(code_dir))
```

#### The whole thing

Now we have the entire, upgraded fabfile that will work with modern Fabric:

```
from invoke import Exit
from invocations.console import confirm

from fabric import task

my_hosts = ["my-server"]

@task
def test(c):
    result = c.local("./manage.py test my_app", warn=True)
    if not result and not confirm("Tests failed. Continue anyway?"):
        raise Exit("Aborting at user request.")

@task
def commit(c):
    c.local("git add -p && git commit")

@task
def push(c):
    c.local("git push")

@task
def prepare_deploy(c):
    test(c)
    commit(c)
    push(c)

@task(hosts=my_hosts)
def deploy(c):
    code_dir = "/srv/django/myproject"
    if not c.run("test -d {}".format(code_dir), warn=True):
        cmd = "git clone user@vcshost:/path/to/repo/.git {}"
        c.run(cmd.format(code_dir))
    c.run("cd {} && git pull".format(code_dir))
    c.run("cd {} && touch app.wsgi".format(code_dir))
```

## 4.7 Development

The Fabric development team is headed by Jeff Forcier, aka `bitprophet`. However, dozens of other developers pitch in by submitting patches and ideas via GitHub issues and pull requests, *IRC* or the mailing list.

### 4.7.1 Get the code

Please see the *Source code checkouts* section of the *Installing* page for details on how to obtain Fabric's source code.

### 4.7.2 Contributing

There are a number of ways to get involved with Fabric:

- **Use Fabric and send us feedback!** This is both the easiest and arguably the most important way to improve the project – let us know how you currently use Fabric and how you want to use it. (Please do try to search the ticket tracker first, though, when submitting feature ideas.)
- **Report bugs or submit feature requests.** We follow contribution-guide.org's guidelines, so please check them out before visiting the ticket tracker.

While we may not always reply promptly, we do try to make time eventually to inspect all contributions and either incorporate them or explain why we don't feel the change is a good fit.

### 4.7.3 Support of older releases

Major and minor releases do not usually mark the end of the previous line or lines of development:

- Recent minor release branches typically continue to receive critical bugfixes, often extending back two or three release lines (so e.g. if 2.4 was the currently active release line, 2.3 and perhaps even 2.2 might get patches).
- Depending on the nature of bugs found and the difficulty in backporting them, older release lines may also continue to get bugfixes – but there's no guarantee of any kind. Thus, if a bug were found in 2.4 that affected 2.1 and could be easily applied, a new 2.1.x version *might* be released.

## 4.8 Troubleshooting

Stuck? Having a problem? Here are the steps to try before you submit a bug report.

- **Make sure you're on the latest version.** If you're not on the most recent version, your problem may have been solved already! Upgrading is always the best first step.
- **Try older versions.** If you're already *on* the latest Fabric, try rolling back a few minor versions (e.g. if on 2.3, try Fabric 2.2 or 2.1) and see if the problem goes away. This will help the devs narrow down when the problem first arose in the commit log.
- **Try switching up your Paramiko.** Fabric relies heavily on the Paramiko library for its SSH functionality, so try applying the above two steps to your Paramiko install as well.

---

**Note:** Fabric versions sometimes have different Paramiko dependencies - so to try older Paramikos you may need to downgrade Fabric as well.

---

- **Make sure Fabric is really the problem.** If your problem is in the behavior or output of a remote command, try recreating it without Fabric involved:
  - Find out the exact command Fabric is executing on your behalf:
    - In 2.x and up, activate command echoing via the `echo=True` keyword argument, the `run.echo` config setting, or the `-e` CLI option.
    - In 1.x, run Fabric with `--show=debug` and look for `run:` or `sudo:` lines.
  - Execute the command in an interactive remote shell first, to make sure it works for a regular human; this will catch issues such as errors in command construction.
  - If that doesn't find the issue, run the command over a non-shell SSH session, e.g. `ssh yourserver "your command"`. Depending on your settings and Fabric version, you may want to use `ssh -T` (disable PTY) or `-t` (enable PTY) to most closely match how Fabric is executing the command.

- **Enable Paramiko-level debug logging.** If your issue is in the lower level Paramiko library, it can help us to see the debug output Paramiko prints. At top level in your fabfile (or in an appropriate module, if not using a fabfile), add the following:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

  This should start printing Paramiko's debug statements to your standard error stream. (Feel free to add more logging kwargs to `basicConfig()` such as `filename='/path/to/a/file'` if you like.)

  Then submit this info to anybody helping you on IRC or in your bug report.

## 4.9 Development roadmap

This document outlines Fabric's intended development path. Please make sure you're reading the latest version of this document, and also see the page about *upgrading* if you are migrating from version 1 to versions 2 or above.

### 4.9.1 Fabric 2 and above

Modern Fabric versions (2+) receive active feature and bugfix development:

- **2.0**: Initial public release, arguably a technology preview and a packaging/upgrade trial. Intent is to act as a jolt for users of 1.x who aren't pinning their dependencies (sorry, folks!), enable installation via PyPI so users don't have to install via Git to start upgrading, and generally get everything above-board and iterating in classic semantic versioning fashion.
- **2.1, 2.2, 2.3, etc**: Implement the most pressing "missing features", including features which were present in 1.0 (see *Upgrading from 1.x* for details on these) as well as any brand new features we've been wanting in 2.x for a while (though most of these will come via Invoke and/or Paramiko releases – see note below for more).
- **3.0, 4.0, etc**: Subsequent major releases will **not** be full-on rewrites as 2.0 was, but will be *small* (feature-release-sized) releases that just happen to contain one or more backwards incompatible API changes. These will be clearly marked in the changelog and reflected in the upgrading documentation.

---

**Note:** Many features that you may use via Fabric will only need development in the libraries Fabric wraps – Invoke and Paramiko – and unless Fabric itself needs changes to match, you can often get new features by upgrading only one of the three. Make sure to check the other projects' changelogs periodically!

---

### 4.9.2 Fabric 1.x

Fabric 1.x has reached a tipping point regarding internal tech debt, lack of testability & ability to make improvements without harming backwards compatibility. As such, the 1.x line now receives bugfixes only. We **strongly** encourage all users to *upgrade* to Fabric 2.x.

## 4.10 Contact

If you've scoured the conceptual and API documentation and still can't find an answer to your question, below are various support resources that should help. We do request that you do at least skim the documentation before posting tickets or mailing list questions, however!

### 4.10.1 Mailing list

The best way to get help with using Fabric is via the fab-user mailing list (currently hosted at `nongnu.org`.) The Fabric developers do their best to reply promptly, and the list contains an active community of other Fabric users and contributors as well.

### 4.10.2 Twitter

Fabric has an official Twitter account, @pyfabric, which is used for announcements and occasional related news tidbits (e.g. "Hey, check out this neat article on Fabric!").

You may also want to follow the principal developer, @bitprophet, for development updates and colorful commentary.

### 4.10.3 Bugs/ticket tracker

To file new bugs or search existing ones, you may visit Fabric's Github Issues page. This does require a (free, easy to set up) Github account.

### 4.10.4 IRC

We maintain a semi-official IRC channel at `#fabric` on Freenode (`irc://irc.freenode.net`) where the developers and other users may be found. As always with IRC, we can't promise immediate responses, but some folks keep logs of the channel and will try to get back to you when they can.