PyTest Skip Markers

Release 1.1.1

VMware, Inc.

Feb 05, 2022

CONTENTS

1 Markers **1**
- 1.1 destructive_test . . . 1
- 1.2 expensive_test . . . 1
- 1.3 skip_if_not_root . . . 2
- 1.4 skip_if_binaries_missing . . . 2
- 1.5 requires_network . . . 2
- 1.6 skip_on_windows . . . 2
- 1.7 skip_unless_on_windows . . . 3
- 1.8 skip_on_linux . . . 3
- 1.9 skip_unless_on_linux . . . 3
- 1.10 skip_on_darwin . . . 4
- 1.11 skip_unless_on_darwin . . . 4
- 1.12 skip_on_sunos . . . 4
- 1.13 skip_unless_on_sunos . . . 4
- 1.14 skip_on_smartos . . . 5
- 1.15 skip_unless_on_smartos . . . 5
- 1.16 skip_on_freebsd . . . 5
- 1.17 skip_unless_on_freebsd . . . 5
- 1.18 skip_on_netbsd . . . 6
- 1.19 skip_unless_on_netbsd . . . 6
- 1.20 skip_on_openbsd . . . 6
- 1.21 skip_unless_on_openbsd . . . 6
- 1.22 skip_on_aix . . . 7
- 1.23 skip_unless_on_aix . . . 7
- 1.24 skip_on_aarch64 . . . 7
- 1.25 skip_unless_on_aarch64 . . . 7
- 1.26 skip_on_spawning_platform . . . 8
- 1.27 skip_unless_on_spawning_platform . . . 8
- 1.28 skip_on_platforms . . . 8
- 1.29 skip_unless_on_platforms . . . 9

2 PyTest Skip Makers Package **11**
- 2.1 Utils . . . 11
  - 2.1.1 PyTest Markers related utilities . . . 11
  - 2.1.2 Platform related utilities . . . 12
  - 2.1.3 Ports related utility functions . . . 14

3 Changelog **15**
- 3.1 [UNRELEASED DRAFT] (2022-02-05) . . . 15
- 3.2 1.1.1 (2022-02-05) . . . 15

3.2.1 Bug Fixes . . . 15
3.3 1.1.0 (2022-01-26) . . . 15
3.3.1 Improvements . . . 15
3.3.2 Trivial/Internal Changes . . . 15
3.4 skip-markers 1.0.0 (2021-10-04) . . . 16
3.4.1 Features . . . 16

Python Module Index **17**

Index **19**

CHAPTER

ONE

MARKERS

1.1 destructive_test

@pytest.mark.**destructive_test**

Skip the test if `--run-destructive` is not passed to pytest on the CLI.

Use this mark when the test does something destructive to the system where the tests are running, for example, adding or removing a user, changing a user password.

Note

Do not use this marker if all the test does is add/remove/change files in the test suite temporary directory

---

```
@pytest.mark.destructive_test
def test_func():
    assert True
```

1.2 expensive_test

@pytest.mark.**expensive_test**

Skip the test if `--run-expensive` is not passed to pytest on the CLI.

Use this test when the test does something expensive(as in monetary expensive), like creating a virtual machine on a cloud provider, etc.

```
@pytest.mark.expensive_test
def test_func():
    assert True
```

1.3 skip_if_not_root

@pytest.mark.**skip_if_not_root**

Skip the test if the user running the test suite is not `root` or `Administrator` on Windows.

```
@pytest.mark.skip_if_not_root
def test_func():
    assert True
```

Look *here* for the full function signature.

1.4 skip_if_binaries_missing

@pytest.mark.**skip_if_binaries_missing**(**binaries*, *check_all=True*, *reason=None*)

Parameters

- **binaries** (*str*) – Any argument passed must be a `str` which is the name of the binary check for presence in the path. Multiple arguments can be passed.
- **check_all** (*bool*) – If `check_all` is `True`, the default, all binaries must exist. If `check_all` is `False`, then only one the passed binaries needs to be found. Useful when, for example, passing a list of python interpreter names(python3.5, python3, python), where only one needs to exist.
- **reason** (*str*) – The skip reason.

Skip tests if binaries are not found in path.

```
@pytest.mark.skip_if_binaries_missing("sshd")
def test_func():
    assert True


@pytest.mark.skip_if_binaries_missing("python3.7", "python3", "python", check_
↪all=False)
def test_func():
    assert True
```

Look *here* for the full function signature.

1.5 requires_network

1.6 skip_on_windows

@pytest.mark.**skip_on_windows**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on windows.

```
@pytest.mark.skip_on_windows
def test_func():
    assert True
```

1.7 skip_unless_on_windows

@pytest.mark.**skip_unless_on_windows**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on windows.

```
@pytest.mark.skip_unless_on_windows
def test_func():
    assert True
```

1.8 skip_on_linux

@pytest.mark.**skip_on_linux**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on linux.

```
@pytest.mark.skip_on_linux
def test_func():
    assert True
```

1.9 skip_unless_on_linux

@pytest.mark.**skip_unless_on_linux**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on linux.

```
@pytest.mark.skip_unless_on_linux
def test_func():
    assert True
```

1.10 skip_on_darwin

@pytest.mark.**skip_on_darwin**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on darwin.

```
@pytest.mark.skip_on_darwin
def test_func():
    assert True
```

1.11 skip_unless_on_darwin

@pytest.mark.**skip_unless_on_darwin**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on darwin.

```
@pytest.mark.skip_unless_on_darwin
def test_func():
    assert True
```

1.12 skip_on_sunos

@pytest.mark.**skip_on_sunos**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on sunos.

```
@pytest.mark.skip_on_sunos
def test_func():
    assert True
```

1.13 skip_unless_on_sunos

@pytest.mark.**skip_unless_on_sunos**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on sunos.

```
@pytest.mark.skip_unless_on_sunos
def test_func():
    assert True
```

1.14 skip_on_smartos

@pytest.mark.**skip_on_smartos**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on smartos.

```
@pytest.mark.skip_on_smartos
def test_func():
    assert True
```

1.15 skip_unless_on_smartos

@pytest.mark.**skip_unless_on_smartos**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on smartos.

```
@pytest.mark.skip_unless_on_smartos
def test_func():
    assert True
```

1.16 skip_on_freebsd

@pytest.mark.**skip_on_freebsd**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on freebsd.

```
@pytest.mark.skip_on_freebsd
def test_func():
    assert True
```

1.17 skip_unless_on_freebsd

@pytest.mark.**skip_unless_on_freebsd**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on freebsd.

```
@pytest.mark.skip_unless_on_freebsd
def test_func():
    assert True
```

1.18 skip_on_netbsd

@pytest.mark.**skip_on_netbsd**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on netbsd.

```
@pytest.mark.skip_on_netbsd
def test_func():
    assert True
```

1.19 skip_unless_on_netbsd

@pytest.mark.**skip_unless_on_netbsd**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on netbsd.

```
@pytest.mark.skip_unless_on_netbsd
def test_func():
    assert True
```

1.20 skip_on_openbsd

@pytest.mark.**skip_on_openbsd**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on openbsd.

```
@pytest.mark.skip_on_openbsd
def test_func():
    assert True
```

1.21 skip_unless_on_openbsd

@pytest.mark.**skip_unless_on_openbsd**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on openbsd.

```
@pytest.mark.skip_unless_on_openbsd
def test_func():
    assert True
```

1.22 skip_on_aix

@pytest.mark.**skip_on_aix**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on aix.

```
@pytest.mark.skip_on_aix
def test_func():
    assert True
```

1.23 skip_unless_on_aix

@pytest.mark.**skip_unless_on_aix**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on aix.

```
@pytest.mark.skip_unless_on_aix
def test_func():
    assert True
```

1.24 skip_on_aarch64

@pytest.mark.**skip_on_aarch64**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on aarch64.

```
@pytest.mark.skip_on_aarch64
def test_func():
    assert True
```

1.25 skip_unless_on_aarch64

@pytest.mark.**skip_unless_on_aarch64**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is running on aarch64.

```
@pytest.mark.skip_unless_on_aarch64
def test_func():
    assert True
```

1.26 skip_on_spawning_platform

@pytest.mark.**skip_on_spawning_platform**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test if test suite is running on a platfor which defaults multiprocessing to `spawn`.

```
@pytest.mark.skip_on_spawning_platform
def test_func():
    assert True
```

1.27 skip_unless_on_spawning_platform

@pytest.mark.**skip_unless_on_spawning_platform**(*reason=None*)

Parameters **reason** (*str*) – The skip reason

Skip test unless the test suite is not running on a platform which defaults multiprocessing to `spawn`.

```
@pytest.mark.skip_unless_on_spawning_platform
def test_func():
    assert True
```

1.28 skip_on_platforms

@pytest.mark.**skip_on_platforms**(***platforms*, *reason=None*)

Parameters

- **windows** (*bool*) – Skip on windows if `True`
- **linux** (*bool*) – Skip on linux if `True`
- **darwin** (*bool*) – Skip on darwin if `True`
- **sunos** (*bool*) – Skip on sunos if `True`
- **smartos** (*bool*) – Skip on smartos if `True`
- **freebsd** (*bool*) – Skip on freebsd if `True`
- **netbsd** (*bool*) – Skip on netbsd if `True`
- **openbsd** (*bool*) – Skip on openbsd if `True`
- **aix** (*bool*) – Skip on aix if `True`
- **aarch64** (*bool*) – Skip on aarch64 if `True`
- **spawning** (*bool*) – Skip on platforms for which multiprocessing defaults to `spawn` if `True`
- **reason** (*str*) – The skip reason

Pass `True` to any of the platforms defined as keyword arguments to skip the test when running on that platform

```
@pytest.mark.skip_on_platforms(windows=True, darwin=True)
def test_func():
    assert True
```

1.29 skip_unless_on_platforms

@pytest.mark.**skip_unless_on_platforms**(***platforms*, *reason=None*)

Parameters

- **windows** (*bool*) – Skip unless on windows if True
- **linux** (*bool*) – Skip unless on linux if True
- **darwin** (*bool*) – Skip unless on darwin if True
- **sunos** (*bool*) – Skip unless on sunos if True
- **smartos** (*bool*) – Skip unless on smartos if True
- **freebsd** (*bool*) – Skip unless on freebsd if True
- **netbsd** (*bool*) – Skip unless on netbsd if True
- **openbsd** (*bool*) – Skip unless on openbsd if True
- **aix** (*bool*) – Skip unless on aix if True
- **aarch64** (*bool*) – Skip on aarch64 if True
- **spawning** (*bool*) – Skip on platforms for which multiprocessing does not default to spawn if True
- **reason** (*str*) – The skip reason

Pass True to any of the platforms defined as keyword arguments to skip the test when not running on that platform

```
@pytest.mark.skip_unless_on_platforms(windows=True, darwin=True)
def test_func():
    assert True
```

CHAPTER

TWO

PYTEST SKIP MAKERS PACKAGE

2.1 Utils

2.1.1 PyTest Markers related utilities

PyTest Markers related utilities.

pytestskipmarkers.utils.markers.**skip_if_not_root**()
Helper function to check for root/Administrator privileges.

Returns: str: The reason of the skip

Return type *Optional*[str]

pytestskipmarkers.utils.markers.**skip_if_binaries_missing**(*binaries*, *check_all=True*, *reason=None*)
Helper function to check for existing binaries.

Args:

binaries (list or tuple): Iterator of binaries to check

check_all (bool): If `check_all` is `True`, the default, all binaries must exist. If `check_all` is `False`, then only one the passed binaries needs to be found. Useful when, for example, passing a list of python interpreter names(python3.5, python3, python), where only one needs to exist.

reason (str): The skip reason.

Returns: str: The reason for the skip. None: Should not be skipped.

Parameters

- **binaries** (*Union[List[str], Tuple[str, ...]]*) –
- **check_all** (*bool*) –
- **reason** (*Optional[str]*) –

Return type *Optional*[str]

pytestskipmarkers.utils.markers.**skip_if_no_local_network**()
Helper function to check for existing local network.

Returns: str: The reason for the skip. None: Should not be skipped.

Return type *Optional*[str]

pytestskipmarkers.utils.markers.**skip_if_no_remote_network**()
Helper function to check for existing remote network(internet).

Returns: str: The reason for the skip. None: Should not be skipped.

Return type *Optional*[str]

pytestskipmarkers.utils.markers.**evaluate_markers**(*item*)
Fixtures injection based on markers or test skips based on CLI arguments.

Parameters **item** (*Item*) –

Return type None

2.1.2 Platform related utilities

Platform related utilities.

pytestskipmarkers.utils.platform.**is_windows**()
Simple function to return if a host is Windows or not.

Return bool Return true on Windows

Return type bool

pytestskipmarkers.utils.platform.**is_linux**()
Simple function to return if a host is Linux or not.

Note for a proxy minion, we need to return something else :return bool: Return true on Linux

Return type bool

pytestskipmarkers.utils.platform.**is_darwin**()
Simple function to return if a host is Darwin (macOS) or not.

Return bool Return true on Darwin(macOS)

Return type bool

pytestskipmarkers.utils.platform.**is_sunos**()
Simple function to return if host is SunOS or not.

Return bool Return true on SunOS

Return type bool

pytestskipmarkers.utils.platform.**is_smartos**()
Simple function to return if host is SmartOS (Illumos) or not.

Return bool Return true on SmartOS (Illumos)

Return type bool

pytestskipmarkers.utils.platform.**is_freebsd**()
Simple function to return if host is FreeBSD or not.

Return bool Return true on FreeBSD

Return type bool

pytestskipmarkers.utils.platform.**is_netbsd**()
Simple function to return if host is NetBSD or not.

Return bool Return true on NetBSD

Return type bool

pytestskipmarkers.utils.platform.**is_openbsd**()
Simple function to return if host is OpenBSD or not.

Return bool Return true on OpenBSD

Return type bool

pytestskipmarkers.utils.platform.**is_aix**()
Simple function to return if host is AIX or not.

Return bool Return true on AIX

Return type bool

pytestskipmarkers.utils.platform.**is_aarch64**()
Simple function to return if host is AArch64 or not.

Return type bool

pytestskipmarkers.utils.platform.**is_spawning_platform**()
Returns `True` if running on a platform which defaults multiprocessing to spawn.

Return type bool

pytestskipmarkers.utils.platform.**on_platforms**(*windows=False*, *linux=False*, *darwin=False*, *sunos=False*, *smartos=False*, *freebsd=False*, *netbsd=False*, *openbsd=False*, *aix=False*, *aarch64=False*, *spawning=False*)
Check to see if we're on one of the provided platforms.

Parameters

- **windows** (*bool*) – When `True`, check if running on Windows.
- **linux** (*bool*) – When `True`, check if running on Linux.
- **darwin** (*bool*) – When `True`, check if running on Darwin.
- **sunos** (*bool*) – When `True`, check if running on SunOS.
- **smartos** (*bool*) – When `True`, check if running on SmartOS.
- **freebsd** (*bool*) – When `True`, check if running on FreeBSD.
- **netbsd** (*bool*) – When `True`, check if running on NetBSD.
- **openbsd** (*bool*) – When `True`, check if running on OpenBSD.
- **aix** (*bool*) – When `True`, check if running on AIX.
- **aarch64** (*bool*) – When `True`, check if running on AArch64.
- **spawning** (*bool*) – When `True`, check if running on a platform which defaults multiprocessing to spawn
- **windows** –
- **linux** –
- **darwin** –
- **sunos** –
- **smartos** –
- **freebsd** –

- **netbsd** –
- **openbsd** –
- **aix** –
- **aarch64** –
- **spawning** –

Return type bool

pytestskipmarkers.utils.platform.**is_fips_enabled**()
Check is FIPS is enabled.

Return bool Return true when enabled

Return type bool

2.1.3 Ports related utility functions

Ports related utility functions.

pytestskipmarkers.utils.ports.**get_unused_localhost_port**(*use_cache=False*)
Return a random unused port on localhost.

Parameters

- **use_cache** (*bool*) – If `use_cache` is `True`, consecutive calls to this function will never return the cached port.
- **use_cache** –

Return type int

pytestskipmarkers.utils.ports.**get_connectable_ports**(*ports*)
Returns a set of the ports where connection was successful.

Parameters **ports** (*Iterable*) – An iterable of ports to try and connect to

Return type set

Returns Returns a set of the ports where connection was successful

CHAPTER

THREE

CHANGELOG

Versions follow Semantic Versioning (<*major*>.<*minor*>.<*patch*>).

Backward incompatible (breaking) changes will only be introduced in major versions with advance notice in the **Deprecations** section of releases.

3.1 [UNRELEASED DRAFT] (2022-02-05)

No significant changes.

3.2 1.1.1 (2022-02-05)

3.2.1 Bug Fixes

- Allow installing on older minior versions of Py3.5. Looking at you Debian. (#10)

3.3 1.1.0 (2022-01-26)

3.3.1 Improvements

- Maintain the skip location under Pytest >= 7.0.x (#7)
- The plugin is now fully typed (#8)

3.3.2 Trivial/Internal Changes

- Reproducible builds
  - Fix copyright headers hook
  - `towncrier` now uses `issue_format` (#7)

3.4 skip-markers 1.0.0 (2021-10-04)

3.4.1 Features

- First public release of the Pytest Skip Markers Plugin

PYTHON MODULE INDEX

u

pytestskipmarkers.utils, 11
pytestskipmarkers.utils.markers, 11
pytestskipmarkers.utils.platform, 12
pytestskipmarkers.utils.ports, 14

INDEX

B

built-in function
    pytest.mark.destructive_test(), 1
    pytest.mark.expensive_test(), 1
    pytest.mark.skip_if_binaries_missing(), 2
    pytest.mark.skip_if_not_root(), 2
    pytest.mark.skip_on_aarch64(), 7
    pytest.mark.skip_on_aix(), 7
    pytest.mark.skip_on_darwin(), 4
    pytest.mark.skip_on_freebsd(), 5
    pytest.mark.skip_on_linux(), 3
    pytest.mark.skip_on_netbsd(), 6
    pytest.mark.skip_on_openbsd(), 6
    pytest.mark.skip_on_platforms(), 8
    pytest.mark.skip_on_smartos(), 5
    pytest.mark.skip_on_spawning_platform(), 8
    pytest.mark.skip_on_sunos(), 4
    pytest.mark.skip_on_windows(), 2
    pytest.mark.skip_unless_on_aarch64(), 7
    pytest.mark.skip_unless_on_aix(), 7
    pytest.mark.skip_unless_on_darwin(), 4
    pytest.mark.skip_unless_on_freebsd(), 5
    pytest.mark.skip_unless_on_linux(), 3
    pytest.mark.skip_unless_on_netbsd(), 6
    pytest.mark.skip_unless_on_openbsd(), 6
    pytest.mark.skip_unless_on_platforms(), 9
    pytest.mark.skip_unless_on_smartos(), 5
    pytest.mark.skip_unless_on_spawning_platform(), 8
    pytest.mark.skip_unless_on_sunos(), 4
    pytest.mark.skip_unless_on_windows(), 3

E

evaluate_markers() (*in module pytestskipmarkers.utils.markers*), 12

G

get_connectable_ports() (*in module pytestskipmarkers.utils.ports*), 14
get_unused_localhost_port() (*in module pytestskipmarkers.utils.ports*), 14

I

is_aarch64() (*in module pytestskipmarkers.utils.platform*), 13
is_aix() (*in module pytestskipmarkers.utils.platform*), 13
is_darwin() (*in module pytestskipmarkers.utils.platform*), 12
is_fips_enabled() (*in module pytestskipmarkers.utils.platform*), 14
is_freebsd() (*in module pytestskipmarkers.utils.platform*), 12
is_linux() (*in module pytestskipmarkers.utils.platform*), 12
is_netbsd() (*in module pytestskipmarkers.utils.platform*), 12
is_openbsd() (*in module pytestskipmarkers.utils.platform*), 13
is_smartos() (*in module pytestskipmarkers.utils.platform*), 12
is_spawning_platform() (*in module pytestskipmarkers.utils.platform*), 13
is_sunos() (*in module pytestskipmarkers.utils.platform*), 12
is_windows() (*in module pytestskipmarkers.utils.platform*), 12

M

module
    pytestskipmarkers.utils, 11
    pytestskipmarkers.utils.markers, 11
    pytestskipmarkers.utils.platform, 12
    pytestskipmarkers.utils.ports, 14

O

on_platforms() (*in module pytestskipmarkers.utils.platform*), 13

P

pytest.mark.destructive_test()
    built-in function, 1
pytest.mark.expensive_test()
    built-in function, 1

pytest.mark.skip_if_binaries_missing()
    built-in function, 2
pytest.mark.skip_if_not_root()
    built-in function, 2
pytest.mark.skip_on_aarch64()
    built-in function, 7
pytest.mark.skip_on_aix()
    built-in function, 7
pytest.mark.skip_on_darwin()
    built-in function, 4
pytest.mark.skip_on_freebsd()
    built-in function, 5
pytest.mark.skip_on_linux()
    built-in function, 3
pytest.mark.skip_on_netbsd()
    built-in function, 6
pytest.mark.skip_on_openbsd()
    built-in function, 6
pytest.mark.skip_on_platforms()
    built-in function, 8
pytest.mark.skip_on_smartos()
    built-in function, 5
pytest.mark.skip_on_spawning_platform()
    built-in function, 8
pytest.mark.skip_on_sunos()
    built-in function, 4
pytest.mark.skip_on_windows()
    built-in function, 2
pytest.mark.skip_unless_on_aarch64()
    built-in function, 7
pytest.mark.skip_unless_on_aix()
    built-in function, 7
pytest.mark.skip_unless_on_darwin()
    built-in function, 4
pytest.mark.skip_unless_on_freebsd()
    built-in function, 5
pytest.mark.skip_unless_on_linux()
    built-in function, 3
pytest.mark.skip_unless_on_netbsd()
    built-in function, 6
pytest.mark.skip_unless_on_openbsd()
    built-in function, 6
pytest.mark.skip_unless_on_platforms()
    built-in function, 9
pytest.mark.skip_unless_on_smartos()
    built-in function, 5
pytest.mark.skip_unless_on_spawning_platform()
    built-in function, 8
pytest.mark.skip_unless_on_sunos()
    built-in function, 4
pytest.mark.skip_unless_on_windows()
    built-in function, 3
pytestskipmarkers.utils
    module, 11
pytestskipmarkers.utils.markers
    module, 11
pytestskipmarkers.utils.platform
    module, 12
pytestskipmarkers.utils.ports
    module, 14

S

skip_if_binaries_missing() (*in module pytestskipmarkers.utils.markers*), 11
skip_if_no_local_network() (*in module pytestskipmarkers.utils.markers*), 11
skip_if_no_remote_network() (*in module pytestskipmarkers.utils.markers*), 11
skip_if_not_root() (*in module pytestskipmarkers.utils.markers*), 11