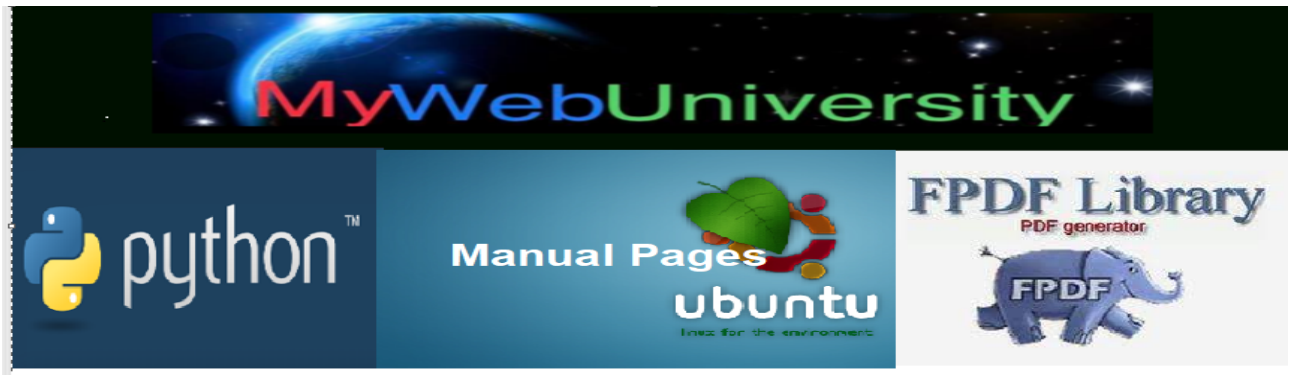

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'pip3-lock.1'

$ man pip3-lock.1

PIP3-LOCK(1) pip PIP3-LOCK(1)

NAME

pip3-lock - description of pip3 lock command

DESCRIPTION

EXPERIMENTAL - Lock packages and their dependencies from:

? PyPI (and other indexes) using requirement specifiers.

? VCS project urls.

? Local project directories.

? Local or remote source archives.

pip also supports locking from "requirements files", which provide an easy way to specify a whole environment to be installed.

The generated lock file is only guaranteed to be valid for the current python version and platform.

USAGE

python -m pip lock [options] [-e] <local project path> ...

python -m pip lock [options] <requirement specifier> [package-index-options] ...

python -m pip lock [options] -r <requirements file> [package-index-options] ...

python -m pip lock [options] <archive url/path> ...

OPTIONS

-o, --output <path>

Lock file name (default=pylock.toml). Use - for stdout.

(environment variable: PIP_OUTPUT)

-r, --requirement <file>

Install from the given requirements file. This option can be used multiple times.

(environment variable: PIP_REQUIREMENT)

-c, --constraint <file>

Constrain versions using the given constraints file. This option can be used multiple times.

(environment variable: PIP_CONSTRAINT)

--no-deps

Don't install package dependencies.

(environment variable: PIP_NO_DEPS, PIP_NO_DEPENDENCIES)

--pre Include pre-release and development versions. By default, pip only finds stable ver? sions.

(environment variable: PIP_PRE)

-e, --editable <path/url>

Install a project in editable mode (i.e. setuptools "develop mode") from a local project path or a VCS url.

(environment variable: PIP_EDITABLE)

--src <dir>

Directory to check out editable projects into. The default in a virtualenv is "<venv path>/src". The default for global installs is "<current dir>/src".

(environment variable: PIP_SRC, PIP_SOURCE, PIP_SOURCE_DIR, PIP_SOURCE_DIRECTORY)

--ignore-requires-python

Ignore the Requires-Python information.

(environment variable: PIP_IGNORE_REQUIRES_PYTHON)

--no-build-isolation

Disable isolation when building a modern source distribution. Build dependencies specified by PEP 518 must be already installed if this option is used.

(environment variable: PIP_NO_BUILD_ISOLATION)

--use-pep517

Use PEP 517 for building source distributions (use --no-use-pep517 to force legacy behaviour).

(environment variable: PIP_USE_PEP517)

--check-build-dependencies

Check the build dependencies when PEP517 is used.

(environment variable: PIP_CHECK_BUILD_DEPENDENCIES)

-C, --config-settings <settings>

Configuration settings to be passed to the PEP 517 build backend. Settings take the form KEY=VALUE. Use multiple --config-settings options to pass multiple keys to the backend.

(environment variable: PIP_CONFIG_SETTINGS)

--no-binary <format_control>

Do not use binary packages. Can be supplied multiple times, and each time adds to the existing value. Accepts either ":all:" to disable all binary packages, ":none:" to empty the set (notice the colons), or one or more package names with commas between them (no colons). Note that some packages are tricky to compile and may fail to in? stall when this option is used on them.

(environment variable: PIP_NO_BINARY)

--only-binary <format_control>

Do not use source packages. Can be supplied multiple times, and each time adds to the existing value. Accepts either ":all:" to disable all source packages, ":none:" to empty the set, or one or more package names with commas between them. Packages with? out binary distributions will fail to install when this option is used on them.

(environment variable: PIP_ONLY_BINARY)

--prefer-binary

Prefer binary packages over source packages, even if the source packages are newer.

(environment variable: PIP_PREFER_BINARY)

--require-hashes

Require a hash to check each requirement against, for repeatable installs. This op? tion is implied when any package in a requirements file has a --hash option.

(environment variable: PIP_REQUIRE_HASHES)

--progress-bar <progress_bar>

Specify whether the progress bar should be used [on, off, raw] (default: on)

(environment variable: PIP_PROGRESS_BAR)

--group <[path:]group>

Install a named dependency-group from a "pyproject.toml" file. If a path is given, the name of the file must be "pyproject.toml". Defaults to using "pyproject.toml" in the current directory.

(environment variable: PIP_GROUP)

--no-clean

Don't clean up build directories.

(environment variable: PIP_NO_CLEAN)

AUTHOR

pip developers

COPYRIGHT