The **opam** package manager for OCaml (command-line)

opam 2.0 · Revision 1 © 2019 OCaml PRO

opam *command* --help	show the manpage for *command*

Non-ambiguous prefixes are accepted
(*e.g.* opam **inst** . --deps for opam **install** . --deps-only).

Installation

Download from:	https://opam.ocaml.org
opam **init**	set up opam , default repository, switch, scripts
opam **init** --bare	create ~/.opam without a compiler switch
opam **init** --reinit -i	reinstall opam scripts (*e.g.* after upgrade)

Run eval $(opam env) when changing switch or prompted, or accept the shell hook setup.

Configuration

opam **config report**	display a summary of the set-up
opam *command* -v[*v*]	print commands being run
opam **config set** *v val*	set switch variable *v*
opam **config set-global** *v val*	set global config variable *v*
opam *command* --root *root*	run opam using *root* as opam root
opam *command* --switch *sw*	run opam on given *sw*
opam **clean**	remove archive cache and artefacts

Switches

opam **switch create** [*name*] *compiler*
  install a new prefix ("switch") with the given *compiler* and select it. *compiler* should be one of ocaml-base-compiler[.*version*], ocaml-system[.*version*], ocaml-variants[.*version*], or --empty.

opam **switch**	list installed switches
opam **switch** *sw*	select the switch *sw*
opam **switch create** *dir* [*compiler*]	install packages defined in *dir* in a new local switch
opam **switch list-available**	list all available compilers

opam **exec** [--switch *sw*] -- *command args*
  run *command args* in the correct environment

The "current switch" is defined by the OPAMSWITCH environment variable, the PWD (for local switches), and the latest selected one.

Allowed URL formats

http:// https:// ftp://	remote archives
ssh:// *file*://	archives or directories
path	file paths (version control is detected)
user@host:path	ssh addresses (using *rsync*)
git:// hg:// darcs://	version control
git+ssh:// hg+https:// git+*file*://	version control with specific transport
git+https://foo.com/git/bar#branch	specific tag, branch, commit, etc.

Packages

opam **install** *pkgs*
  pkgs are package names, *pkg.version*, constraints "*pkg*>=*version*"

opam **install** --show	only print a list of actions
opam **install** --dry-run	simulate everything
opam **install** *pkgs* --best-effort	don't fail on impossible requests
opam **update** [--all]	update repositories and package sources
opam **upgrade** [*pkgs*]	bring installed packages to latest version
opam **remove** *pkgs*	uninstall packages
opam **remove** --auto	uninstall no longer needed dependencies
opam **reinstall** *pkgs*	recompile and reinstall packages
opam **source** *pkg* [--dev]	download package source
opam **reinstall** --list-pending	show pending recompilations
opam **reinstall** --forget-pending	at your own risk

Exploring

opam **list**	list installed packages
opam **list** --resolve *pkg*	list a *sufficient* set of dependencies to install *pkg*
opam **list** [--rec] --required-by *pkg*	list dependencies of *pkg*
opam **list** [--rec] --depends-on *pkg*	list packages depending on *pkg*
opam **list** --roots	exclude automatically-installed dependencies
opam **list** --external *pkg*	list external *pkg* dependencies
opam **list** --owns-file *file*	find package owning *file*
opam **show** *pkg* [--field=*flds*]	show package details
opam **show** *pkg* --raw	show package opam file
opam **show** *pkg* --list-files	list all files belonging to *pkg*
opam **var** *v*	print value of opam variable
opam **config list** [*pkg*]	list variables [of package *pkg*]

Package pinning

opam **install** *dir*	pin and install packages from the sources and definitions at *dir*
opam **pin** *pkg version*	pin *pkg* to given version
opam **pin** *pkg*[.*version*] *url*	pin *pkg*[1] to *url* (can be a dir) and install
opam **pin** *url*	pin using package definitions at *url*
opam **pin** --dev *pkg*	pin known package to its source repo
opam **pin** [--short]	list pinned packages
opam **pin remove** *pkgs*|*dir*	unpin packages
opam **pin edit** *pkg*	tweak package definition

pin commands also install/remove unless -n is specified.

[1]If not using *pkg.version*, version is defined by opam file, directory name, or latest known version.

Project development

Working with local pins

opam **install** *pkg*|*dir* --deps-only
  just install all the pre-requisites
opam **install** *pkg*|*dir* --working-dir
  bypass VCS, take all uncommitted changes
opam **install** *pkg*|*dir* --inplace-build
  process build and install directly in the source
opam **install** *pkg*|*dir* --assume-built
  directly run install commands from the source
opam **lint** *pkg*|*dir*|opamfile
  check the style of a package definition

Sharing a dev setup

opam **lock** *pkg* --direct-only
  generate an opam.locked file with version-strict dependencies
opam **lock** *pkg*
  generate an opam.locked file with a fixed dependency tree
opam **install** *dir*|*pkg* --locked
  install, reproducing the same state as described by the locked file
opam **switch export**|**import** *file*|-
  switch state (compiler, installed packages, pins...) save/restore

Configuring remotes

opam **repository** [--all]
  list defined repositories (current switch, or all)
opam **switch create** --repos default,custom=*url* ...
  create a switch with repositories default, and newly defined custom
opam **repository add** *name url* --dont-select
  define repository *name* at *url*
opam **repository add** *name* [*url*]
  use *name* in the current switch
opam **repository add** *name* [*url*] --set-default
  use *name* for newly created switches
opam **repository add** *name* [*url*] --all-switches
  use *name* for all existing switches
opam **repository add** *name* [*url*] --rank=-1
  use *name* with lowest priority
opam **repository set-url** *name url*
  change repository url
opam **repository set-repos** foo,bar
  redefine the repos selections for the current switch

The definition for *pkg.version* is taken from the highest ranking repository.

The **opam** package manager for OCaml (packaging and tools)

Package definition files

Full specification:	`http://opam.ocaml.org/doc/Manual.html#opam`
In source:	`opam`, or `pkg.opam`, or `opam/pkg.opam`
In a package repository:	`packages/pkg/pkg.version/opam`

```
opam-version: "2.0"
name: "project"
version: "0.1"
synopsis: "One-line description"
description: """
Longer description
"""
maintainer: "Name <email>"
authors: ["Name <email>"]
license: "SPDX license"  # see https://spdx.org/licenses/
homepage: "https://project.org"
bug-reports: "https://gitfoo.net/project/issues"
dev-repo: "git+https://gitfoo.net/project.git"
depends: ["ocaml"
          "ocamlfind" {<= "1.8"}
          "odoc" {with-doc & >= "1.0"}]
# with a regular ./configure - make
build: [["./configure" "--prefix=%{prefix}%"]
        [make]]
install: [make "install"]
# with dune (no 'install:' needed)
depends: ["dune" {>= "1.10"}] # add to your other 'depends:'
build: ["dune" "build" "-p" name "-j" jobs]
```

Some optional fields

`tags: ["org:foo" "examples"]`	for package sorting
`depopts: [deps]`	optional dependencies
`substs: ["foo"]`	expand file `"foo"` from `"foo.in"`
`patches: ["f.patch" {os = "macos"}]`	conditional patches
`run-test: [cmds]`	only when running with `--with-test`
`pin-depends: [["pkg.version" "url"]]`	when pinned, pin also these
`conflicts: [deps]`	anti-dependencies
`available: condition`	pre-requirements
`build-env: [CC = "foo"]`	custom build/install environment
`extra-source "fname" {src: "url" checksum: "sha256=..."}`	additional downloads
`post-messages: """message""" {condition}`	print to the user after install

When in a repository (not in-source):

```
url {
  src: "url"                 archive URL (or VCS, in custom repos)
  checksum: "sha512=XXX"     supported: md5, sha256, sha512
}
```

Expressions

Variables are strings, booleans or undefined values.

postfix conditions	`[make "opt" {condition} "foo"] {condition}`
dependencies	`("p1" {>= "0.5" & != "0.7" & condition} | "p2")`
version ordering	`"1.02"` = `"1.2"` < `"1.12"` < `"2.0~"` < `"2.0"`
comparisons	`var = "value"`, `var != ""`, `"0.1" <= var`
interpolation	`"can be %{var}% or %{bool-var?foo:bar}%"`
undefined	`(?undef)` is `false`, `(undef | true)` is `true`

`_:var` is `pkg:var` for the current package

Some useful variables:

Strings

`name`, `version`	current package name, version allowed *e.g.* as **depends:** `["foo" {= version}]`
`lib`	this is `"%{prefix}%/lib"`
`pkg:lib`	this is `"%{prefix}%/pkg/lib"`
`arch`, `os`, `os-distribution`, `os-family`, `os-version`	system detection

Booleans

`pkg:dev`	*pkg* was not built from a release archive
`with-test`	tests have been enabled (package-specific)
`with-doc`	documentation has been enabled (package-specific)
`build`	(only in depends) don't recompile when changed
`post`	(only in depends) not needed at build time

Run `opam` **`var`** for more

External dependencies

`name: "conf-gtk3"`	by convention, use a "conf-" prefix
`depexts: ["libgtk-3-dev"] {os-family = "debian"}`	define system package dependencies
`flags: conf`	package without install, for polling the system

Related commands:

`opam` **`list`** `-e --resolve pkg`	print requirements of *pkg* on this system
`opam` **`depext`** `pkg`	handles requirements of *pkg* (plugin)

Publishing

Through Github pull-requests to the official repository at
`https://github.com/ocaml/opam-repository`

Automatically, using the `opam-publish` plugin:

`opam` **`publish`** `url`	publish from hosted source archive (plugin)
`opam` **`publish`** `[dir]`	publish latest tag from detected Github origin

Repository administration

To be run from the root of an opam repository:

`opam` **`admin list`**	list packages
`opam` **`admin cache`**	download all archives to cache
`opam` **`admin index`**	generate an index (needed for HTTP)
`opam` **`admin lint`**	lint all packages
`opam` **`admin filter`** `patterns`	only keep matching packages
`opam` **`admin add-constraint`** `"pkg<=3"`	add a version constraint to all dependencies towards *pkg*