NAME

cargo — The Rust package manager

SYNOPSIS

cargo [*options*] *command* [*args*]
cargo [*options*] **−−version**
cargo [*options*] **−−list**
cargo [*options*] **−−help**
cargo [*options*] **−−explain** *code*

DESCRIPTION

This program is a package manager and build tool for the Rust language, available at <https://rust−lang.org>.

COMMANDS

Build Commands

cargo−bench(1)
Execute benchmarks of a package.

cargo−build(1)
Compile a package.

cargo−check(1)
Check a local package and all of its dependencies for errors.

cargo−clean(1)
Remove artifacts that Cargo has generated in the past.

cargo−doc(1)
Build a package's documentation.

cargo−fetch(1)
Fetch dependencies of a package from the network.

cargo−fix(1)
Automatically fix lint warnings reported by rustc.

cargo−run(1)
Run a binary or example of the local package.

cargo−rustc(1)
Compile a package, and pass extra options to the compiler.

cargo−rustdoc(1)
Build a package's documentation, using specified custom flags.

cargo−test(1)
Execute unit and integration tests of a package.

Manifest Commands

cargo−add(1)
Add dependencies to a **Cargo.toml** manifest file.

cargo−generate−lockfile(1)
Generate **Cargo.lock** for a project.

cargo−info(1)

Display information about a package in the registry. Default registry is crates.io.

cargo−locate−project(1)
Print a JSON representation of a **Cargo.toml** file's location.

cargo−metadata(1)
Output the resolved dependencies of a package in machine−readable format.

cargo−pkgid(1)
Print a fully qualified package specification.

cargo−remove(1)
Remove dependencies from a **Cargo.toml** manifest file.

cargo−tree(1)
Display a tree visualization of a dependency graph.

cargo−update(1)
Update dependencies as recorded in the local lock file.

cargo−vendor(1)
Vendor all dependencies locally.

Package Commands

cargo−init(1)
Create a new Cargo package in an existing directory.

cargo−install(1)
Build and install a Rust binary.

cargo−new(1)
Create a new Cargo package.

cargo−search(1)
Search packages in crates.io.

cargo−uninstall(1)
Remove a Rust binary.

Publishing Commands

cargo−login(1)
Save an API token from the registry locally.

cargo−logout(1)
Remove an API token from the registry locally.

cargo−owner(1)
Manage the owners of a crate on the registry.

cargo−package(1)
Assemble the local package into a distributable tarball.

cargo−publish(1)
Upload a package to the registry.

cargo−yank(1)

Remove a pushed crate from the index.

General Commands

cargo−help(1)
Display help information about Cargo.

cargo−version(1)
Show version information.

OPTIONS

Special Options

−V, **−−version**
Print version info and exit. If used with **−−verbose**, prints extra information.

−−list
List all installed Cargo subcommands. If used with **−−verbose**, prints extra information.

−−explain *code*
Run **rustc −−explain CODE** which will print out a detailed explanation of an error message (for example, **E0004**).

Display Options

−v, **−−verbose**
Use verbose output. May be specified twice for "very verbose" output which includes extra output such as dependency warnings and build script output. May also be specified with the **term.verbose** *config value* <https://doc.rust−lang.org/cargo/reference/config.html>.

−q, **−−quiet**
Do not print cargo log messages. May also be specified with the **term.quiet** *config value* <https://doc.rust−lang.org/cargo/reference/config.html>.

−−color *when*
Control when colored output is used. Valid values:

- **auto** (default): Automatically detect if color support is available on the terminal.
- **always**: Always display colors.
- **never**: Never display colors.

May also be specified with the **term.color** *config value* <https://doc.rust−lang.org/cargo/reference/config.html>.

Manifest Options

−−locked
Asserts that the exact same dependencies and versions are used as when the existing **Cargo.lock** file was originally generated. Cargo will exit with an error when either of the following scenarios arises:

- The lock file is missing.
- Cargo attempted to change the lock file due to a different dependency resolution.

It may be used in environments where deterministic builds are desired, such as in CI pipelines.

−−offline
Prevents Cargo from accessing the network for any reason. Without this flag, Cargo will stop with an error if it needs to access the network and the network is not available. With this flag, Cargo will

attempt to proceed without the network if possible.

Beware that this may result in different dependency resolution than online mode. Cargo will restrict itself to crates that are downloaded locally, even if there might be a newer version as indicated in the local copy of the index. See the **cargo−fetch**(1) command to download dependencies before going offline.

May also be specified with the **net.offline** *config value* <https://doc.rust−lang.org/cargo/reference/config.html>.

−−frozen
Equivalent to specifying both **−−locked** and **−−offline**.

Common Options

+*toolchain*
If Cargo has been installed with rustup, and the first argument to **cargo** begins with **+**, it will be interpreted as a rustup toolchain name (such as **+stable** or **+nightly**). See the *rustup documentation* <https://rust−lang.github.io/rustup/overrides.html> for more information about how toolchain overrides work.

−−config *KEY=VALUE* or *PATH*
Overrides a Cargo configuration value. The argument should be in TOML syntax of **KEY=VALUE**, or provided as a path to an extra configuration file. This flag may be specified multiple times. See the *command−line overrides section* <https://doc.rust−lang.org/cargo/reference/config.html#command−line−overrides> for more information.

−C *PATH*
Changes the current working directory before executing any specified operations. This affects things like where cargo looks by default for the project manifest (**Cargo.toml**), as well as the directories searched for discovering **.cargo/config.toml**, for example. This option must appear before the command name, for example **cargo −C path/to/my−project build**.

This option is only available on the *nightly channel* <https://doc.rust−lang.org/book/appendix−07−nightly−rust.html> and requires the **−Z unstable−options** flag to enable (see *#10098* <https://github.com/rust−lang/cargo/issues/10098>).

−h, **−−help**
Prints help information.

−Z *flag*
Unstable (nightly−only) flags to Cargo. Run **cargo −Z help** for details.

ENVIRONMENT

See *the reference* <https://doc.rust−lang.org/cargo/reference/environment−variables.html> for details on environment variables that Cargo reads.

EXIT STATUS

- **0**: Cargo succeeded.
- **101**: Cargo failed to complete.

FILES

~/.cargo/
Default location for Cargo's "home" directory where it stores various files. The location can be changed with the **CARGO_HOME** environment variable.

$CARGO_HOME/bin/
Binaries installed by **cargo−install**(1) will be located here. If using *rustup* <https://rust−lang.github.io/rustup/>, executables distributed with Rust are also located here.

$CARGO_HOME/config.toml
The global configuration file. See *the reference* <https://doc.rust−lang.org/cargo/reference/config.html> for more information about configuration files.

.cargo/config.toml
Cargo automatically searches for a file named **.cargo/config.toml** in the current directory, and all parent directories. These configuration files will be merged with the global configuration file.

$CARGO_HOME/credentials.toml
Private authentication information for logging in to a registry.

$CARGO_HOME/registry/
This directory contains cached downloads of the registry index and any downloaded dependencies.

$CARGO_HOME/git/
This directory contains cached downloads of git dependencies.

Please note that the internal structure of the **$CARGO_HOME** directory is not stable yet and may be subject to change.

EXAMPLES

1. Build a local package and all of its dependencies:

```
cargo build
```

2. Build a package with optimizations:

```
cargo build −−release
```

3. Run tests for a cross−compiled target:

```
cargo test −−target i686−unknown−linux−gnu
```

4. Create a new package that builds an executable:

```
cargo new foobar
```

5. Create a package in the current directory:

```
mkdir foo && cd foo
cargo init .
```

6. Learn about a command's options and usage:

```
cargo help clean
```

BUGS

See <https://github.com/rust−lang/cargo/issues> for issues.

SEE ALSO

rustc(1), **rustdoc**(1)