Cargo Rust package manager

CONCEPTS

Use cargo check while editing: it skips code generation and is much faster than build.

Workspace root Cargo.toml, Cargo.lock, target/

Package one Cargo.toml, a crate or more

Target lib, bin, example, test, bench

A workspace shares one Cargo.lock and one target/ directory

Cargo.toml is what you edit; Cargo.lock records the exact resolved versions. Commit the lock for binaries.

Arguments after -- go to the program or test harness, not to Cargo.

PROJECTS

cargo new *<name>*
Create a binary package

cargo init
Make the current directory a package

cargo run
Build and run the binary

cargo run --bin *<name>*
Run a specific binary

cargo clean
Delete the target directory

cargo doc --open
Build docs and open them

cargo install *<crate>*
Install a binary crate

BUILD & CHECK

cargo build
Compile in the dev profile

cargo build --release
Compile with optimisations

cargo check
Type-check without building

cargo check --all-targets
Also check tests and examples

TEST & LINT

cargo test
Run unit, integration and doc tests

cargo test *<filter>*
Run tests whose name contains text

cargo test -- --nocapture
Show output of passing tests

cargo fmt
Format with rustfmt

cargo clippy
Lint with Clippy

DEPENDENCIES

cargo add *<crate>*
Add a dependency

cargo add --dev *<crate>*
Add a dev-dependency

cargo remove *<crate>*
Remove a dependency

cargo update
Update Cargo.lock within semver

cargo tree
Show the dependency tree

cargo tree -i *<crate>*
Who depends on this crate?

FEATURES

cargo build --features *<list>*
Enable features

cargo build --no-default-features
Turn off the default feature set

WORKSPACES

cargo build --workspace
Build every member

cargo test -p *<pkg>*
Test one package

cargo build -p *<pkg>*
Build one package

PUBLISHING

cargo publish --dry-run
Run all checks, upload nothing

cargo publish
Upload to crates.io