Cargo Rust package manager

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

CONCEPTS

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

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

PROJECTS (cont.)

`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

`cargo build --target <triple>`
Cross-compile

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

TEST & LINT (cont.)

`cargo test --lib`
Unit tests only

`cargo test --doc`
Doc tests only

`cargo test --test <name>`
One integration test file

`cargo test --no-fail-fast`
Keep going after a failure

`cargo fmt`
Format with rustfmt

`cargo fmt --check`
Fail if code is unformatted

`cargo clippy`
Lint with Clippy

`cargo clippy -- -D warnings`
Treat lints as errors

DEPENDENCIES

`cargo add <crate>`
Add a dependency

`cargo add <crate>@<ver>`
Add with a version requirement

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

DEPENDENCIES (cont.)

`cargo add <crate> -F <feat>`
Add with a feature enabled

`cargo remove <crate>`
Remove a dependency

`cargo update`
Update Cargo.lock within semver

`cargo update <crate>`
Update only one crate

`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

`cargo build --all-features`
Enable every feature

FEATURES (cont.)

Cargo.toml features

```
[features]
default = ["json"]
json = ["dep:serde_json"]
fast = []

[dependencies]
serde_json = { version =
"1", optional = true }
```

WORKSPACES

`cargo build --workspace`
Build every member

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

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

`cargo run -p <pkg>`
Run a member's binary

PUBLISHING

`cargo login`
Store a registry token

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

`cargo publish`
Upload to crates.io