rustup Rust toolchain manager

CONCEPTS

> rustup is a toolchain multiplexer: rustc and cargo in ~/.cargo/bin are proxies that forward to the active toolchain.

Channel stable / beta / nightly
↓ `install`
Toolchain channel + date + host
↓ `component add`
Components clippy, rustfmt, ...
↓ `target add`
Targets std per platform

Which toolchain wins (first match)

Source	Example
1. +toolchain argument	`cargo +beta`
2. env RUSTUP_TOOLCHAIN	`=beta`
3. directory override	`override set 1.80`
4. rust-toolchain.toml	`channel = ...`
5. default	`default stable`

> Overrides and toolchain files are looked up from the current directory upwards; the nearest one wins.

INSTALL & UPDATE

`rustup update`
Update installed toolchains and rustup

`rustup update stable`
Update only one toolchain

`rustup check`
Check for updates without installing

`rustup self update`
Update rustup itself only

TOOLCHAINS

`rustup toolchain install <toolchain>`
Install a toolchain

`rustup toolchain list`
List installed toolchains

`rustup toolchain uninstall <toolchain>`
Remove a toolchain

`rustup toolchain install <toolchain> --profile minimal`
Install without docs and extras

TOOLCHAINS (cont.)

`rustup toolchain install <toolchain> -c <component> -t <target>`
Install with components and targets

`rustup toolchain install 1.80`
Install a specific version

`rustup set profile <profile>`
Profile for future installs

DEFAULT & OVERRIDES

`rustup default <toolchain>`
Set the global default toolchain

`rustup default`
Print the current default

`rustup override set <toolchain>`
Pin a toolchain for this directory

`rustup override unset`
Remove this directory's override

`rustup show`
Active toolchain, installed ones, why

`rustup show active-toolchain`
Print only the active toolchain

`rustup which <command>`
Path of the binary that would run

> For teams, commit a rust-toolchain.toml instead of using overrides: it travels with the repo.

RUN WITH A TOOLCHAIN

`cargo +<toolchain> <command>`
Run cargo with another toolchain once

`rustup run <toolchain> <command>`
Run any command in a toolchain

> A +toolchain argument wins over everything, including rust-toolchain.toml.

COMPONENTS

`rustup component list`
Available and installed components

COMPONENTS (cont.)

`rustup component list --installed`
Only installed components

`rustup component add <component>`
Add a component

`rustup component add <component> --toolchain <toolchain>`
Add to another toolchain

`rustup component add rust-src`
Std source for IDEs and build-std

`rustup component add rust-analyzer`
Language server for editors

`rustup component add clippy rustfmt`
Linter and formatter

`rustup doc --std`
Open the offline std docs

TARGETS

`rustup target list`
All available targets

`rustup target list --installed`
Targets you have installed

`rustup target add <target>`
Install std for a target

`rustup target add <target> --toolchain <toolchain>`
Add to another toolchain

`cargo build --target <target>`
Build for that target

`rustup target add wasm32-unknown-unknown`
WebAssembly

> A target only brings std; you usually still need a linker, configured per target in Cargo's config.

PIN A PROJECT

rust-toolchain.toml

```
[toolchain]
channel = "1.80"
components = ["rustfmt", "clippy"]
targets = ["wasm32-unknown-unknown"]
profile = "minimal"
```