rustup Rust toolchain manager

START HERE

`rustup` **update**
Update all toolchains

`rustup` **component add** *<name>*
Add clippy, rustfmt and more

`rustup` **toolchain install** *<toolchain>*
Install a toolchain

`rustup` **target add** *<target>*
Cross-compile support

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

`rustup` **show**
What is active, and why?

`rustup` **override set** *<toolchain>*
Pin this directory

cargo +*<toolchain>* *<cmd>*
One-off toolchain

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

CONCEPTS

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

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
Name form: <channel>[-<date>][-<host>], e.g. nightly-2026-01-15 or stable-msvc.

`rustup` **toolchain list**
List installed toolchains

`rustup` **toolchain uninstall** *<toolchain>*
Remove a toolchain

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

`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
minimal = rustc, rust-std, cargo; default adds rust-docs, rustfmt, clippy; complete is not recommended.

DEFAULT & OVERRIDES

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

DEFAULT & OVERRIDES (cont.)

`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
Works for any proxy: rustc +nightly --version.

RUN WITH A TOOLCHAIN (cont.)

cargo +nightly fmt
Example: nightly-only rustfmt options

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

RUSTUP_TOOLCHAIN=*<toolchain>* *<command>*
Pick a toolchain via environment

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

COMPONENTS

`rustup` **component list**
Available and installed components

`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 remove** *<component>*
Remove a component

COMPONENTS (cont.)

`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
Needs the rust-docs component (in the default profile).

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

TARGETS (cont.)

`rustup` **target add x86_64-unknown-linux-musl**
Static Linux binaries

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"
```

Missing toolchain, components and targets are installed automatically the next time you run cargo or rustc there.

RUSTUP_DIST_SERVER
Download from a mirror

RECIPES

Try your crate on nightly

```
rustup toolchain install
nightly --profile minimal
cargo +nightly test
# nothing global changed
```

Minimal CI toolchain

```
rustup toolchain install stable
--profile minimal -c clippy -c
rustfmt --no-self-update
rustup default stable
```