The Rust ecosystem

crates.io is the registry. Read the README, the license, and the recent releases before you add a crate. Download counts are a weak signal.

docs.rs builds the documentation for a published version. Use the version menu that matches `Cargo.lock`. The standard library docs are the first place to look for a type you hope already exists. `cargo doc --open` builds those pages for your crate and its dependencies, on your machine. lib.rs is another index of the same crates, grouped by category.

Clippy is the linter. This course runs it with `-D warnings`. rustfmt is the formatter. rust-analyzer is the language server in the editor. `cargo add` writes the dependency line.

cargo audit checks known vulnerabilities against the RustSec advisory database. cargo deny checks licenses, bans, and those advisories in CI.

Useful extras around the everyday loop: cargo-binstall installs prebuilt cargo tools quickly. bacon is the main save-and-rerun feedback loop (the `php artisan serve`-style habit). watchexec can do the same with `watchexec -e rs -- cargo test`. cargo-chef caches Cargo dependencies in Docker layers, which pairs with Running in Docker.

This course uses edition 2024: a new edition ships every three years and is opt-in, so old crates keep compiling. Stable Rust ships a new release every six weeks, with beta and nightly beside it.

Read a crate's guide before its trait bounds. The module page on docs.rs is the API. A trait-bound error usually means the guide's import or feature flag is missing. Axum runs on Tokio. The JSON types are serde. You swap one of them by reading the replacement, then running the tests.

Crate	What it's for	Chapter

clap	Command line arguments	CLI and files
serde	Derive formats from Rust types	Getting started
serde_json	JSON text	CLI and files
csv	CSV files	CLI and files
anyhow	Errors for an application	Error handling
thiserror	An error type for a library	Error handling
rayon	Data parallel iterators	Concurrency
tokio	The async runtime the API runs on	Concurrency
axum	The HTTP service	Building an API
sqlx	Postgres queries	Building an API
tower-http	CORS and request tracing	Building an API
tracing	Structured log events	Building an API
tracing-subscriber	Printing those events	Building an API
tower	One-shot requests in API tests	Building an API
http-body-util	Reading a response body in a test	Building an API
proptest	Property tests	Testing

criterion	Timing a hot path with `cargo bench`	Testing
assert_cmd	Running the CLI in a test	CLI and files
leptos	The browser UI	WebAssembly
wasm-bindgen	The Rust and JavaScript boundary	WebAssembly
gloo-net	HTTP from the browser	WebAssembly

> **Watch for:** Adding a crate because the download count is high. Read the README, the license, and the guide for the version you pin.

> **Idiomatic Rust:** Upgrade one crate at a time, then `cargo test` and clippy.

Side by side

	PHP	Rust
Packages	Composer and Packagist	Cargo and crates.io
Docs for the language	php.net	The standard library docs and docs.rs
Lint and format	PHP_CodeSniffer, PHP-CS-Fixer	clippy and rustfmt

Key takeaways

- crates.io, docs.rs, lib.rs, and the standard library docs are where you look a crate up.
- clippy, rustfmt, rust-analyzer, `cargo add`, cargo audit, and cargo deny are the tools around Cargo.
- The table on this page is every crate this course depended on.

What's next: Where to go from here.