Installing Rust

Getting started · Installing Rust

In PHP

You install PHP from the operating system, Homebrew, or php.new, then you install Composer beside it. `php -v` and `composer -V` tell you what you actually have. Rust's equivalent of both tools is one installer: rustup. It installs the compiler `rustc` and the build tool `cargo`, and it can keep more than one toolchain on the machine.

In Rust

On macOS or Linux, install from the official script. Read the prompt. The default is stable, which is what you want.

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Windows uses rustup-init.exe. Install the **MSVC** toolchain and the Visual Studio Build Tools workload **Desktop development with C++** when the installer offers them — without that workload, `cargo build` fails with a linker error (`link.exe` / `lld-link` not found). After it finishes, open a **new** PowerShell or Terminal window so `PATH` includes `%USERPROFILE%\.cargo\bin`.

For the Postgres and Docker chapters later, use **WSL2** (Ubuntu) and run those lessons inside the Linux environment. rustup also works inside WSL with the same curl installer as macOS/Linux.

On Linux you need a C linker before the first build (`build-essential` on Debian/Ubuntu, or `gcc` / `clang` on other distros). On macOS run `xcode-select --install` once so the Command Line Tools provide a linker.

Codespaces: open the companion examples repo with the devcontainer so Rust, wasm targets, sqlx-cli, and Postgres are already there.

Confirm all three. `rustup` is the toolchain manager. `rustc` is the compiler. `cargo` is the build tool and package manager, in the same way Composer is yours.

```
 rustup show
rustc --version
cargo --version
```

`rustup show` should list an active toolchain of `stable-<your-target>` (for example `stable-aarch64-apple-darwin` or `stable-x86_64-unknown-linux-gnu`). If a version prints, the install worked.

Editor: any editor is fine. VS Code plus the `rust-analyzer` extension is the usual setup. Set `"rust-analyzer.check.command": "clippy"` so the editor runs the same lints as `cargo clippy`. For JetBrains, use RustRover or the JetBrains Rust plugin (they use their own engine, not rust-analyzer). Neovim works with rust-analyzer as the language server.

You do not need nightly to follow this course. The next lessons use stable only.

> **Watch for:** copying `rustup default nightly` from an old post. That changes the compiler for every project on the machine. The nightly lesson shows the per-directory command instead.

> **Idiomatic Rust:** install with rustup, then trust `rustup show`, not a distro package named `rustc` that can lag years behind.

Side by side

	PHP	Rust
Install	Homebrew, phpenv, or php.new	rustup
The package tool	Composer, installed beside PHP	`cargo`, installed with rustup

	PHP	Rust
Check	`php -v` and `composer -V`	`rustup show`, then `rustc` and `cargo`

Windows (PowerShell)

Confirm the toolchain the same way:

```
 rustup show
rustc --version
cargo --version
```

Environment variables for later chapters (Postgres, sqlx) look like this in PowerShell:

```
 $env:DATABASE_URL = "postgres://shop:shop@127.0.0.1:54329/shop"
$env:SQLX_OFFLINE = "true"
```

If `cargo build` fails with a linker error after a fresh install, the MSVC Build Tools are missing — re-run the Visual Studio Installer and enable **Desktop development with C++**, then open a new shell.

Try it

Run the three commands above. `cargo --version` and `rustc --version` should report the same release.

Key takeaways

- rustup installs `rustc` and `cargo` together.
- `rustup show` is the check that you are on stable.
- rust-analyzer should check with clippy.
- On Windows, rustup-init.exe plus the MSVC C++ workload; use WSL2 for Docker/Postgres chapters.

What's next: Hello Rust