# **Pre-Coding Essentials (Component: rust-toolchain.toml, Version/FormulaID: VM-ENGINE v0) — 2/89**

## **1) Goal & Success**

Goal: Pin a single Rust toolchain so builds/tests are reproducible across machines.

Success: rustc --version matches the pinned version; cargo build --locked/cargo test --locked pass on Windows/macOS/Linux for x86-64 & arm64.

## **2) Scope**

In scope: channel/version pin, required components, optional targets.

Out of scope: per-crate deps/profiles (in Cargo.toml), network policy (.cargo/config.toml).

## **3) Inputs → Outputs**

Inputs: desired Rust **stable** version (exact), component list (rustfmt, clippy), target triples.

Outputs: rust-toolchain.toml recognized by rustup; rustup show displays pinned toolchain.

## **4) Entities/Tables**

## **5) Variables**

## **6) Functions**

(None.)

## **7) Algorithm Outline**

Set channel = "1.xx.x" (exact version; no toolchain drift).

Add components = ["rustfmt","clippy"].

Optionally add targets = [ "x86\_64-unknown-linux-gnu", "aarch64-unknown-linux-gnu", "x86\_64-apple-darwin", "aarch64-apple-darwin", "x86\_64-pc-windows-msvc", "aarch64-pc-windows-msvc" ] if CI builds cross-OS.

Avoid profile overrides here; keep profiles in Cargo.toml.

## **8) State Flow**

rustup reads toolchain file → installs exact toolchain → cargo uses it.

## **9) Determinism & Numeric Rules**

Determinism via exact version pin; no numeric rules.

## **10) Edge Cases & Failure Policy**

Nightly-only features in crates → **fail** (we require stable).

Missing component → rustup installs it on first run; CI must cache toolchain.

Cross-targets unavailable on host → skip adding to targets unless required.

## **11) Test Checklist**

rustc --version equals pinned version.

rustup toolchain list shows the pinned toolchain (default for workspace).

cargo fmt -- --version and cargo clippy -V succeed.

Build CLI on all three OSes (host builds): cargo build --locked -p vm\_cli passes.