

Updated on 02/04/2026

Register

# Jetbrains RustRover Training

3 days (21 hours)

## Overview

JetBrains RustRover helps you code in Rust faster and with greater peace of mind thanks to code analysis, smart navigation, and integrated debugging. This training focuses on concrete use cases: starting a Cargo project, refactoring safely, diagnosing borrow checker errors, and industrializing your workflow.

The goal of our training is to make you self-sufficient in RustRover so you can develop, test, and maintain Rust applications with a tooled environment: assisted editing, execution, testing, profiling, and Git integration. You will learn how to configure the IDE for your context (workspace, toolchain, clippy, rustfmt) and how to use inspections to improve quality.

The approach is decidedly practical: guided workshops, step-by-step demos, and correction exercises.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

## Objectives

- Configure RustRover, Cargo, and the Rust toolchain.
- Navigate a codebase efficiently (symbols, usages, types).
- Run and debug binaries and tests with breakpoints.
- Apply rustfmt, clippy, and inspections to refactor.
- Automate workflow via run configurations and Git.

## Target audience

- Beginner to intermediate Rust developers.
- System/embedded developers who want to equip their IDE.
- Back-end developers adopting Rust in production.

## Prerequisites

- Basic programming skills (types, functions, structures).
- Basic knowledge of Rust (ownership/borrow, modules, Cargo).
- Understanding of unit testing.
- Minimal experience with Git.

## Technical prerequisites

- Minimum 8 GB RAM, 16 GB recommended.
- Windows, macOS, or Linux (x64/ARM depending on machine).
- Stable Rust installed (rustup), Cargo, rustfmt, clippy.
- JetBrains RustRover installed, terminal access, and Git.

## Jetbrains RustRover training program

[Day 1 - Morning]

### Getting started with RustRover and setting up a Rust environment

- Install RustRover, Rust toolchain (rustup), and components (clippy, rustfmt)
- Configure a Cargo project: workspace, profiles, environment variables
- Overview of the IDE: Project, Editor, Run, Terminal, Search Everywhere
- Configuring Rust analysis: rust-analyzer, inspections, quick fixes
- Hands-on workshop: Create a Cargo workspace and validate the toolchain (build, fmt, clippy).

[Day 1 - Afternoon]

### Daily productivity: navigation, refactoring, and execution

- Efficient navigation: symbols, usages, hierarchy, bookmarks, file structure
- Useful refactorings: rename, extract function, inline, move, change signature
- Execute and debug: Run/Debug configurations, breakpoints, watch, evaluate
- Test management: targeted execution, filters, coverage, logs, and output
- Hands-on workshop: Refactoring a module, adding tests, and debugging a failure case.

[Day 2 - Morning]

## Code quality: inspections, formatting, linting, and documentation

- Setting up rustfmt: rules, format on save, formatting actions
- Using clippy: lint levels, quick fixes, targeted deletion/allow
- Understanding RustRover inspections: errors, warnings, intentions, and quick fixes
- Generating and viewing documentation: rustdoc, doc comments, API navigation
- Hands-on workshop: Clean up a codebase (fmt + clippy), document an API, and validate in local CI.

### [Day 2 - Afternoon]

## Dependency management, advanced builds, and Git integration

- Mastering Cargo.toml: features, optional deps, dev-dependencies, patch
- Profiling the build: debug/release profiles, incremental, target dir, cache
- Cargo tools in the IDE: run configurations, tasks, frequent commands
- Git workflow: commit, amend, interactive rebase, conflict resolution in the IDE
- Hands-on workshop: Adding a dependency with features, optimizing a build, and managing a PR with conflicts.

### [Day 3 - Morning]

## Advanced debugging, testing, and performance diagnostics

- Advanced debugging: conditional breakpoints, logpoints, attach to a process
- Rust testing: unit/integration, parameterized tests, parallel execution, and filtering
- Error analysis: backtraces, panic, structured display, and logs
- First steps in performance: profiles, benchmarks, hotspot detection (Cargo tools)
- Hands-on workshop: Diagnosing an intermittent bug with conditional breakpoints and stabilizing via testing.

### [Day 3 - Afternoon]

## Industrialization: templates, local CI, and good team practices

- Standardizing a project: Cargo templates, crate structure, module conventions
- Automating quality: hooks, tasks (fmt/clippy/test), pre-commit checks
- Preparing a "portable" CI: scripts, reproducible commands, environment variables
- Good IDE practices for teams: shared settings, exclusions, recommended inspections
- Hands-on workshop: Build a CI-ready project skeleton (fmt + clippy + tests) and share it with the group.

## Companies involved

This training is aimed at both individuals and companies, large or small, wishing to train their teams in new advanced IT technology or to acquire specific professional knowledge or modern methods.

## Placement at the start of training

The placement test at the start of the training course complies with Qualiopi quality criteria. Once they have finalized their registration, learners receive a self-assessment questionnaire that allows us to assess their estimated level of proficiency in different types of technology, as well as their expectations and personal objectives for the upcoming training course, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could be problematic for the monitoring and smooth running of the training session.

## Teaching methods

Practical training: 60% practical, 40% theory. Training materials distributed in digital format to all participants.

## Organization

The course alternates between theoretical input from the trainer, supported by examples and reflection sessions, and group work.

## Validation

At the end of the session, a multiple-choice questionnaire is used to verify that the skills have been correctly acquired.

## Certification

A certificate will be issued to each trainee who has completed the entire training course.