Rustup 不仅仅是个“更新器”，它还管理着 Rust 工具链，里面包含官方的内部组件，比如 rust-analyzer. 管理整个 Rust 工具链，安装匹配版本的 rustc、cargo 和所有其他组件。

Rustc 是个编译器，它把 Rust 代码编译成原生的机器码。Rustc 是编译器本身，使用一个更底层的接口。它的选项并不总是稳定的，而且可能很难正确设置，所以 cargo 旨在为你处理这个问题。在 Rust 中，你通常不会直接调用编译器。

Cargo 管理 Rust 包（crates）—— 第三方、用户空间的二进制文件。cargo 管理各种 crate 及其依赖项以及编译 rust 代码时的构建选项。它下载包（crates），解决版本和依赖关系图，并按正确的顺序在它们上面调用编译器。它还确保构建脚本和过程宏得到编译并适当地应用。

Cargo 是一个用于管理 Rust 项目的工具。它有几个功能。首先，Cargo 负责包管理。如果你想从互联网上使用第三方库，你可以使用 Cargo 轻松处理。

其次，Cargo 有许多方便的功能。你可以调用 cargo check 对你的代码进行快速的语法和类型检查， cargo build 编译你的代码， cargo test 测试你的代码， cargo doc 从你的文档中生成漂亮的 HTML 文档，以及 cargo run 运行你的代码。Cargo 会自动为你处理所有步骤。例如，如果你运行 cargo run，它会检查自上次更改代码以来你是否已经编译了代码。如果你的代码需要编译或重新编译，Cargo 会先这样做，然后运行你的代码。最后，使用 Cargo 设置你的项目意味着你可以在线发布它，其他人可以使用 Cargo 包含它。

再来解释一下 Cargo，首先，如果有人写了一个库，其他程序员可以用，那么有很多种方法可以分发它——比如，在作者自己的网站上提供下载，或者类似的方式。在一些语言生态系统中（包括 Rust），最重要的分发方式变成了一个仓库系统，这里有一个集中收集所有重要库的地方，作者也会上传到那里（也）。

所以，如果你写一个程序，也使用了几个库，并且你想在没有 Cargo 或类似工具的情况下完成所有操作:

对于每个库，你都要搜索从哪里获取它并下载它。

你还需要记住时不时地检查是否有更新的库版本，例如修复了安全漏洞或其他问题

在你的程序中使用每个库之前，都需要先编译它。可能需要一些说明，因为它可能并不简单（依赖于其他库，可能不是 Rust 的东西，特殊的编译器标志，可以编译或不编译的可选功能，等等）。

你可能还想在某些情况下修改这个过程，例如优化大小而不是速度，或者其他什么。

对于你自己的程序，你需要决定传递给 rustc 的哪些标志来编译它，涉及很多方面。优化、输出类型，等等。外部库也需要包含在内，然后你需要再次调用链接器，并使用一些需要匹配的标志。

每次编译时，你至少需要调用两个很长的命令行，其中包含许多标志和路径。并且要注意，如果你想将一个文件移动到另一个目录，你可能需要调整你的构建步骤。

使用 Cargo：

你开始为你的程序编写一个配置文件。在其中，有类似“我想为小程序而不是速度进行优化”和“我需要库 foozurz，至少是版本 44”的内容。

要编译你的项目，你将运行“cargo build”。

Cargo 正在做的事情，例如：

从中央仓库下载库

检查是否有更新的版本

构建每个库，将其自己的配置（以及可能用适当的编程语言编写的扩展构建代码，如果配置不够强大）与你自己的偏好（大小优化）相结合

也编译你自己的程序，遵守你的配置。（同样，如果你有配置文件无法处理的特定需求，仍然可以这样说“将标志 -abc 传递给 rustc”，但通常配置文件就足够了）。

它处理文件位置等，也使 rustc 调用适应它们的变化

它也会自动调用链接器