Red Hat Developer Tools 1

使用 Rust 1.71.1 Toolset

安装和使用 Rust 1.71.1 Toolset

Last Updated: 2023-12-14

Red Hat Developer Tools 1 使用 Rust 1.71.1 Toolset

安装和使用 Rust 1.71.1 Toolset

Jacob Valdez
jvaldez@redhat.com

法律通告

摘要

Rust Toolset 是 Red Hat Enterprise Linux (RHEL)操作系统的开发人员提供的红帽产品。使用本指南了解 Rust Toolset 的概述，了解如何调用和使用不同版本的 Rust 工具，并通过更深入的信息查找资源。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 RUST TOOLSET

Rust Toolset 是 Red Hat Enterprise Linux (RHEL)的开发人员提供的红帽产品。它为 Rust 编程语言提供 Rust **c** 编译器、Rust 软件包管理器 Cargo、Rustfmt 格式化工具和所需的库。

对于 Red Hat Enterprise Linux 8，Rust Toolset 作为模块提供。Rust Toolset 作为 Red Hat Enterprise Linux 9 的软件包提供。

1.1. RUST TOOLSET 组件

以下组件作为 Rust Toolset 的一部分提供：

Name	版本	描述
rust	1.71.1	LLVM 的 Rust 编译器前端。
cargo	1.71.1	Rust 的构建系统和依赖项管理器。
rustfmt	1.71.1	用于自动格式化 Rust 代码的工具。

1.2. RUST TOOLSET 兼容性

Rust Toolset 在以下构架中为 Red Hat Enterprise Linux 8 和 Red Hat Enterprise Linux 9 提供：

- **AMD 和 Intel 64 位**
- **64-bit ARM**
- **IBM Power Systems, Little Endian**
- **64-bit IBM Z**

1.3. 安装 RUST TOOLSET

完成以下步骤以安装 Rust Toolset，包括所有开发和调试工具以及依赖软件包。请注意，Rust

Toolset 依赖于 LLVM Toolset。

先决条件

- 已安装所有可用的 Red Hat Enterprise Linux 更新。

流程

在 Red Hat Enterprise Linux 8 中，运行以下命令安装 Rust-toolset 模块：

```
# yum module install rust-toolset
```

在 Red Hat Enterprise Linux 9 中，运行以下命令安装 Rust-toolset 软件包：

```
# dnf install rust-toolset
```

1.4. 安装 RUST 文档

Rust 编程语言书 作为可安装的文档提供。

先决条件

- 已安装 Rust Toolset。
  如需更多信息，请参阅安装 Rust Toolset。

流程

要安装 Rust-doc 软件包，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

  ```
  # yum install rust-doc
  ```

  您可以在以下路径下找到 *Rust 编程语言* 书： /usr/share/doc/rust/html/index.html。
  您可以在以下路径中找到所有 Rust 代码软件包的 API 文档：
  /usr/share/doc/rust/html/std/index.html。

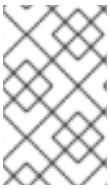
- 在 Red Hat Enterprise Linux 9 中：

```
# dnf install rust-doc
```

您可以在以下路径下找到 *Rust 编程语言* 书：/usr/share/doc/rust/html/index.html。
您可以在以下路径中找到所有 Rust 代码软件包的 API 文档：
/usr/share/doc/rust/html/std/index.html。

1.5. 安装 CARGO 文档

Cargo, Rust 的 Package Manager 书作为 Cargo 的可安装文档。



注意

在 Rust Toolset 1.71 中，cargo-doc 软件包包含在 Rust -doc 软件包中。

先决条件

- 已安装 Rust Toolset。
如需更多信息，请参阅安装 Rust Toolset。

流程

- 要安装 cargo-doc 软件包，请运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
# yum install cargo-doc
```

您可以在以下路径下找到 *Cargo, Rust 的 Package Manager* 书：
/usr/share/doc/cargo/html/index.html。

  - 在 Red Hat Enterprise Linux 9 中：

```
# dnf install cargo-doc
```

您可以在以下路径下找到 *Cargo, Rust 的 Package Manager* 书：/usr/share/doc/cargo/html/index.html。

1.6. 其他资源

- 有关 Rust 编程语言的更多信息，请参阅官方 Rust 文档。

第 2 章 CARGO 构建工具

cargo 是 Rust 编译器 Rust c 的构建工具和 前端，以及软件包和依赖项管理器。它允许 Rust 项目声明具有特定版本要求的依赖关系，解析完整依赖项图、下载软件包和构建，以及测试整个项目。

Rust Toolset 与 Cargo 1.71.1 一起发布。

2.1. CARGO 目录结构和文件放置

Cargo 构建工具使用设置惯例来定义在 Cargo 软件包中目录结构和文件放置。运行 cargo new 命令会为清单和项目文件生成软件包目录结构和模板。默认情况下，它还在软件包根目录中初始化新的 Git 存储库。

对于二进制程序，Cargo 会创建一个目录 *project_name*，其中包含一个名为 Cargo.toml 的文本文件，以及一个包含名为 main.rs 的文本文件的子目录 src。

其他资源

- 有关 Cargo 目录结构的更多信息，请参阅 Cargo Book - 软件包布局。
- 有关 Rust 代码机构的信息，请参阅 Rust 编程语言 - 使用软件包、类别和模块管理项目。

2.2. 创建 RUST 项目

根据 Cargo 约定，创建一个新的 Rust 项目。有关 Cargo 约定的更多信息，请参阅 Cargo 目录结构和文件放置。

流程

运行以下命令来创建 Rust 项目：

- 在 Red Hat Enterprise Linux 8 中：

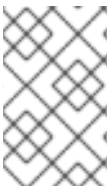
```
$ cargo new --bin <project_name>
```

- 将 *<project_name>* 替换为您的项目名称。

- 在 Red Hat Enterprise Linux 9 中：

```
$ cargo new --bin <project_name>
```

- 将 *<project_name>* 替换为您的项目名称。



注意

要编辑项目代码，请编辑主可执行文件 main.rs，并将新源文件添加到 src 子目录。

其他资源

- 有关配置项目并添加依赖项的详情，请参考 配置 Rust 项目依赖项。

2.3. 创建 RUST 库项目

完成以下步骤，使用 Cargo 构建工具创建 Rust 库项目。

流程

要创建 Rust 库项目，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

```
$ cargo new --lib <project_name>
```

- 将 *<project_name>* 替换为 Rust 项目的名称。

- 在 Red Hat Enterprise Linux 9 中：

```
$ cargo new --lib <project_name>
```

- 将 *<project_name>* 替换为 Rust 项目的名称。

注意

要编辑项目代码，请编辑 src 子目录中的源文件 lib.rs。

其他资源

- 使用软件包、策略和模块管理增长项目

2.4. 构建 RUST 项目

使用 Cargo 构建工具构建 Rust 项目。cargo 解析项目的所有依赖项，下载缺少的依赖项，并使用 Rust c 编译器编译它。

默认情况下，项目以调试模式构建和编译。有关以发行模式编译项目的详情，请参考 以发行模式构建 Rust 项目。

先决条件

- 现有的 Rust 项目。
  有关如何创建 Rust 项目的详情，请参考 创建 Rust 项目。

流程

- 要构建由 Cargo 管理的 Rust 项目，请在项目目录中运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
$ cargo build
```

  - 在 Red Hat Enterprise Linux 9 中：

```
$ cargo build
```

- 要验证您的 Rust 程序是否可以在您不需要构建可执行文件时构建，请运行：

```
$ cargo check
```

2.5. 以发行版本模式构建 RUST 项目

使用 Cargo 构建工具以发行版本模式构建 Rust 项目。发行模式用于优化源代码，因此可以增加编译时间，同时确保编译的二进制文件可以更快地运行。使用此模式生成适合发布和生产的优化的工件。cargo 解析项目的所有依赖项，下载缺少的依赖项，并使用 Rust c 编译器编译它。

有关以调试模式编译项目的详情，请参考 构建 Rust 项目。

先决条件

- 现有的 Rust 项目。
  有关如何创建 Rust 项目的详情，请参考 创建 Rust 项目。

流程

- 要以 release 模式构建项目，请运行：
  - 在 Red Hat Enterprise Linux 8 中：

    ```
    $ cargo build --release
    ```

  - 在 Red Hat Enterprise Linux 9 中：

    ```
    $ cargo build --release
    ```

- 要在不需要构建可执行文件时验证您的 Rust 程序是否可以构建，请运行：

```
$ cargo check
```

2.6. 运行 RUST 程序

使用 Cargo 构建工具运行 Rust 项目。cargo 首先重建您的项目，然后运行生成的可执行文件。如果在

开发过程中使用，cargo run 命令可以正确地解析输出路径，独立于构建模式。

先决条件

- 构建的 Rust 项目。
  有关如何构建 Rust 项目的详情，请参考 构建 Rust 项目。

流程

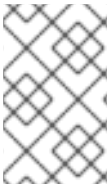
要通过 Cargo 运行作为项目管理的 Rust 程序，请在项目目录中运行：

- 在 Red Hat Enterprise Linux 8 中：

```
$ cargo run
```

- 在 Red Hat Enterprise Linux 9 中：

```
$ cargo run
```



注意

如果您的程序还没有构建，Cargo 会在运行之前构建您的程序。

2.7. 测试 RUST 项目

使用 Cargo 构建工具测试您的 Rust 程序。cargo 首先重新构建您的项目，然后运行项目中找到的测试。请注意，您只能测试免费、monomorphic 并且不使用任何参数的功能。函数返回类型必须是 （） 或 Result< （）, E>，其中 E: Error。

默认情况下，Rust 项目在 debug 模式中测试。有关在发行版本模式中测试您的项目的详情，请参考以发行模式测试 Rust 项目。

先决条件

- 构建的 Rust 项目。
  有关如何构建 Rust 项目的详情，请参考 构建 Rust 项目。

流程

- 在功能前面添加 test 属性 #[test]。
- 要为由 Cargo 管理的 Rust 项目运行测试，请在项目目录中运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
$ cargo test
```

  - 在 Red Hat Enterprise Linux 9 中：

```
$ cargo test
```

其他资源

- 有关在 Rust 项目中执行测试的更多信息，请参阅 Rust Reference - 测试属性。

2.8. 以发行版本模式测试 RUST 项目

使用 Cargo 构建工具以发行版本模式测试您的 Rust 程序。发行模式用于优化源代码，因此可以增加编译时间，同时确保编译的二进制文件可以更快地运行。使用此模式生成适合发布和生产的优化的工件。cargo 首先重新构建您的项目，然后运行项目中找到的测试。请注意，您只能测试免费、monomorphic 并且不使用任何参数的功能。函数返回类型必须是 （） 或 Result< （）, E>，其中 E: Error。

有关以调试模式测试项目的信息，请参阅 测试 Rust 项目。

先决条件

- 构建的 Rust 项目。
  有关如何构建 Rust 项目的详情，请参考 构建 Rust 项目。

流程

- 在功能前面添加 test 属性 #[test]。

- 要为在发行模式的 Cargo 管理的 Rust 项目运行测试，请在项目目录中运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
$ cargo test --release
```

  - 在 Red Hat Enterprise Linux 9 中：

```
$ cargo test --release
```

其他资源

- 有关在 Rust 项目中执行测试的更多信息，请参阅 Rust Reference - 测试属性。

2.9. 配置 RUST 项目依赖项

使用 Cargo 构建工具配置 Rust 项目的依赖项。要指定由 Cargo 管理的项目的依赖项，请编辑项目目录中的 Cargo.toml 文件并重新构建您的项目。cargo 下载 Rust 代码软件包及其依赖项，将其存储在本地，构建包括依赖项代码软件包在内的所有项目源代码，并运行生成的可执行文件。

先决条件

- 构建的 Rust 项目。
  有关如何构建 Rust 项目的详情，请参考 构建 Rust 项目。

流程

1. 在项目目录中，打开文件 Cargo.toml。
2. 移到标记的 [dependencies] 部分。
   每个依赖项都以以下格式列在新行中：

```
crate_name = version
```

   Rust 代码软件包称为 crates。

3. 编辑依赖项。

4. 运行以下命令重建项目：

   - 在 Red Hat Enterprise Linux 8 中：

     ```
     $ cargo build
     ```

   - 在 Red Hat Enterprise Linux 9 中：

     ```
     $ cargo build
     ```

5. 使用以下命令运行项目：

   - 在 Red Hat Enterprise Linux 8 中：

     ```
     $ cargo run
     ```

   - 在 Red Hat Enterprise Linux 9 中：

     ```
     $ cargo run
     ```

其他资源

- 有关配置 Rust 依赖项的更多信息，请参阅 Cargo Book - 指定依赖项。

2.10. 为 RUST 项目构建文档

使用 Cargo 工具从标记为提取的源代码中的注释生成文档。请注意，仅针对公共功能、变量和成员提取文档注释。

先决条件

- 构建的 Rust 项目。
  有关如何构建 Rust 项目的详情，请参考 构建 Rust 项目。

- 配置的依赖项。
有关配置依赖项的更多信息，请参阅配置 Rust 项目依赖项。

流程

- 要标记提取的注释，请使用三个斜杠 ///，并将您的注释放在要记录的行首。
cargo 支持您评论的 Markdown 语言。

- 要使用 Cargo 构建项目文档，请在项目目录中运行：

  - 在 Red Hat Enterprise Linux 8 中：

```
$ cargo doc --no-deps
```

  - 在 Red Hat Enterprise Linux 9 中：

```
$ cargo doc --no-deps
```

生成的文档位于 .target/doc 目录中。

其他资源

- 有关使用 Cargo 构建文档的更多信息，请参阅 Rust 编程语言 - 制作有用的文档注释。

2.11. 在 RED HAT ENTERPRISE LINUX 8 和 RED HAT ENTERPRISE LINUX 9 BETA 上使用 RUST 将代码编译到 WEBASSEMBLY 二进制文件

完成以下步骤以安装 WebAssembly 标准库。

先决条件

- 已安装 Rust Toolset。
如需更多信息，请参阅安装 Rust Toolset。

流程

- 要安装 WebAssembly 标准库，请运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
# yum install rust-std-static-wasm32-unknown-unknown
```

  - 在 Red Hat Enterprise Linux 9 中：

```
# dnf install rust-std-static-wasm32-unknown-unknown
```

- 要将 WebAssembly 与 Cargo 搭配使用，请运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
# cargo <command> --target wasm32-unknown-unknown
```

    将 <command > 替换为您要运行的 Cargo 命令。

  - 在 Red Hat Enterprise Linux 9 中：

```
# cargo <command> --target wasm32-unknown-unknown
```

    将 <command > 替换为您要运行的 Cargo 命令。

其他资源

- 有关 WebAssembly 的更多信息，请参阅官方 Rust 和 WebAssembly 文档或 Rust 和 WebAssembly 书。

2.12. 供应商 RUST 项目依赖项

创建 Rust 项目依赖项的本地副本，以便离线重新发布并使用 Cargo 构建工具重复使用。此流程称为供应商项目依赖项。厂商的依赖项，包括用于在 Windows 操作系统上构建项目的 Rust 代码软件包位于 供

应商 目录中。Cargo 可以使用供应商的依赖项，而无需连接到互联网。

先决条件

- 构建的 Rust 项目。
  有关如何构建 Rust 项目的详情，请参考 构建 Rust 项目。
- 配置的依赖项。
  有关配置依赖项的更多信息，请参阅配置 Rust 项目依赖项。

流程

要使用 Cargo 来厂商您的 Rust 项目，请在项目目录中运行：

- 在 Red Hat Enterprise Linux 8 中：

```
$ cargo vendor
```

- 在 Red Hat Enterprise Linux 9 中：

```
$ cargo vendor
```

2.13. 其他资源

- 有关 Cargo 的更多信息，请参阅官方 Cargo 指南。
- 要显示 Rust Toolset 中包含的手册页，请运行：
  - 对于 Red Hat Enterprise Linux 8：

```
$ man cargo
```

  - 对于 Red Hat Enterprise Linux 9：

```
$ man cargo
```

第 3 章 RUSTFMT 格式化工具

使用 Rustfmt 格式工具，您可以自动格式化 Rust 程序的源代码。您可以将 rusftmt 用作独立工具，也可以使用 Cargo。

3.1. 安装 RUSTFMT

完成以下步骤以安装 Rustfmt 格式化 工具。

先决条件

- 已安装 Rust Toolset。
  如需更多信息，请参阅安装 Rust Toolset。

流程

运行以下命令来安装 Rust fmt ：

- 在 Red Hat Enterprise Linux 8 中：

```
# yum install rustfmt
```

- 在 Red Hat Enterprise Linux 9 中：

```
# dnf install rustfmt
```

3.2. 使用 RUSTFMT 作为独立工具

使用 Rustfmt 作为独立工具来格式化 Rust 源文件及其所有依赖项。作为替代方案，在 Cargo 构建工具中使用 Rustfmt。如需更多信息，请参阅使用带有 Cargo 的 Rustfmt。

先决条件

- 现有的 Rust 项目。
  有关如何创建 Rust 项目的详情，请参考 创建 Rust 项目。

流程

要使用 Rustfmt 作为独立 工具 格式化 Rust 源文件，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

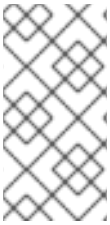
```
$ rustfmt <source-file>
```

  - 将 *<source_file >* 替换为源文件的名称。
另外，您可以将 &lt ;*source_file*> 替换为标准输入。Rust fmt 随后在标准输出中提供其输出。

- 在 Red Hat Enterprise Linux 9 中：

```
$ rustfmt <source-file>
```

  - 将 *<source_file >* 替换为源文件的名称。
另外，您可以将 &lt ;*source_file*> 替换为标准输入。Rust fmt 随后在标准输出中提供其输出。



注意

默认情况下，Rust fmt 修改受影响的文件，而不显示详情或创建备份。要显示详情并创建备份，请使用 **--write-mode** *值运行 Rustfmt*。

3.3. 使用带有 CARGO 构建工具的 RUSTFMT

使用带有 Cargo 的 Rustfmt 工具格式化 Rust 源文件及其所有依赖项。
作为替代方案，使用 Rust fmt 作为独立工具。如需更多信息，请参阅使用 Rustfmt 作为独立工具。

先决条件

- 现有的 Rust 项目。
有关如何创建 Rust 项目的详情，请参考 创建 Rust 项目。

流程

要格式化 Cargo 代码软件包中的所有源文件，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

```
$ cargo fmt
```

- 在 Red Hat Enterprise Linux 9 中：

```
$ cargo fmt
```

注意

要更改 Rust fmt 格式选项，请在 项目目录中创建配置文件 Rustfmt.toml，并将您的配置添加到该文件中。

3.4. 其他资源

- 要显示 Rustfmt 的帮助页面， 请运行：
  - 在 Red Hat Enterprise Linux 8 中：

```
$ rustfmt --help
```

  - 在 Red Hat Enterprise Linux 9 中：

```
$ rustfmt --help
```

- 要配置 Rust fmt 工具，请在项目目录中创建 Rustfmt .toml 配置文件，并将您的配置添加到该文件中。您可以在 Configuration .md 文件中找到配置选项。
  - 在 Red Hat Enterprise Linux 8 中，您可以在以下路径中找到它：

    /usr/share/doc/rustfmt/Configurations.md

  - 在 Red Hat Enterprise Linux 9 中，您可以在以下路径中找到它：

/usr/share/doc/rustfmt/Configurations.md

第 4 章 RHEL 8 上带有 RUST TOOLSET 的容器镜像

在 RHEL 8 中，您可以使用 Containerfiles 在 Red Hat Universal Base Images (UBI)容器之上构建自己的 Rust Toolset 容器镜像。

4.1. 在 RHEL 8 中创建 RUST TOOLSET 的容器镜像

在 RHEL 8 中，Rust Toolset 软件包是 Red Hat Universal Base Images (UBI)存储库的一部分。要将容器大小保持小，请只安装单个软件包，而不是整个 Rust Toolset。

先决条件

- 现有的 Containerfile。
  有关创建 Containerfiles 的更多信息，请参阅 Dockerfile 参考 页面。

流程

- 访问 Red Hat Container Catalog。
- 选择 UBI。
- 点 Get this image 并按照说明进行操作。
- 要创建包含 Rust Toolset 的容器，请在 Containerfile 中添加以下行：

```
FROM registry.access.redhat.com/ubi8/ubi:latest

RUN yum install -y rust-toolset
```

- 要创建仅包含单个软件包的容器镜像，请在 Containerfile 中添加以下行：

```
RUN yum install <package-name>
```

- 将 < *package_name* > 替换为您要安装的软件包的名称。

4.2. 其他资源

- 如需有关 Red Hat UBI 镜像的更多信息，请参阅使用容器镜像。

- 如需有关 Red Hat UBI 存储库的更多信息，请参阅 通用基础镜像(UBI)：镜像、存储库、软件包和源代码。

第 5 章 RUST 1.71.1 TOOLSET 中的更改

Rust Toolset 已从 RHEL 8 和 RHEL 9 上的 1.66.1 版本更新至 1.71.1。

主要变更包括：

- 多个生产者(mpsc)通道的新实施，以提高性能。
- 新的 Cargo 稀疏 索引协议，用于更有效地使用 crates.io registry。
- 新的 OnceCell 和 OnceLock 类型用于一次性值初始化。
- 新的 C-unwind ABI 字符串，可在 Foreign Function Interface (FFI)范围内使用强制取消卷。

有关更新的详情，请查看上游发布公告系列：

- 宣布 Rust 1.67.0
- 宣布 Rust 1.68.0
- 宣布 Rust 1.69.0
- 宣布 Rust 1.70.0
- 宣布 Rust 1.71.0