Cargo

Sin cargo

```
$ rustc main.rs
```

- **Compila** un archivo con código en lenguaje Rust.

```
$ ./main
```

- **Ejecuta** el archivo compilado.

¿Que es cargo?

- Es un administrador de dependencias, por lo tanto se encarga de descargar y/o construir las dependencias.
- Es un sistema de compilación, por lo tanto se encarga de construir el archivo ejecutable.
- Permite mejorar la organización de los proyectos.

Gestión de proyectos con cargo

Creación

```
$ cargo new nuevo_proyecto
```

- Crea un nuevo proyecto
- Carpetas y archivos creados
  - **src**: Contiene el código del proyecto, el directorio superior es para archivos README, información de licencias, archivos de configuración y cualquier otra cosa que no esté relacionada al código
  - **.git**: Carpeta del VCS (Sistema de Control del Versiones)
  - **.gitignore**: Archivo oculto del VCS::Git que excluye el versionado de ciertos archivos declarados.

    - **target/debug**: Ubicación donde cargo almacena el resultado de la compilación de una versión en desarrollo del programa.
    - **target/release**: Ubicación donde cargo almacena el resultado de la compilación de una versión estable y optimizada del programa
    - **Cargo.toml**: Archivo de configuración del proyecto. Utiliza el formato TOML que es un formato de configuración cargo. Estructura del archivo:
        - **Encabezado *[package]***: Declaraciones de configuración para el paquete.
        - **Encabezado *[dependencies]***: Enumeración de las dependencias / crates.

```
$ cargo new --help
```

- Muestra otras opciones disponibles de creación

Compilación

```
$ cargo build
```

- Compila el proyecto y construye el archivo ejecutable
- Carpetas y archivos creados o actualizados después de ejecutar el comando:
    - **/target/debug**: Contiene el archivo ejecutable.
    - **Cargo.lock**: Realiza un seguimiento de las versiones exactas de las dependencias del proyecto (no requiere volver a descubrir todas las versiones de las dependencias), se auto-mantienen con cargo.

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

- Compila una versión estable del proyecto y construye el archivo ejecutable optimizado.
- La compilación **es más lenta**, pero el ejecutable es más rápido. Por lo general es para un ambiente de producción o entrega final.
- Carpetas y archivos creados o actualizados después de ejecutar el comando:
    - **/target/release**: Contiene el archivo ejecutable.
    - **Cargo.lock**

Ejecución

```
$ cargo run
```

- Compila el proyecto, construye el archivo ejecutable y ejecuta el archivo ejecutable de la carpeta `/target/debug`

Verificación

```
$ cargo check
```

- Verifica si el código del proyecto se compila sin errores.
- Este comando es más rápido que el comando `$ cargo build` porque no produce un ejecutable, pero sí construye el proyecto.

Actualización de dependencias

```
$ cargo update
```

- Actualiza las pequeñas versiones (v0.0.x) de las dependencias.
- Actualiza el archivo cargo.lock.

Versión de cargo

```
$ cargo --version
```

- Muestra la versión actual de cargo instalada

Documentación del lenguaje Rust

```
$ cargo doc --open
```

- Abre la documentación del lenguaje Rust.