Piecing Together Rust

It's more than just writing code

By Tarun Pothulapati

About Me

- I’m Tarun Pothulapati 👋
- Engineer @ Buoyant i.e Makers of Linkerd.
- Prev: Intern at CNCF, working on Linkerd.
- Develops primarily in Golang but got started with Rust this year.
- Contributing to Rust OSS projects like tracing, etc recently.
- Also, started biking recently and plan to do more of it.
- Find me at tarun.xyz

Installation

Getting Rust

Rustup

- Toolchain (single installation of the Rust compiler) Multiplexer.
- Installs and manages multiple Rust toolchains.
- Each tools usually consists of multiple components.
- Some components involve rustc, cargo, rustfmt, rust-std, rustdoc, rls, rust-analyzer, clippy, miri, rust-src, etc.
- Components availability may vary between different releases and toolchains.
- Custom toolchains are also supported to have local builds, etc.
- One type of toolchains is Channels.
- 3 different cycles:
    - **Stable**: 6 weeks
    - **Beta**: Released before a stable
    - **Nightly**: Daily


example on master [?] is v0.1.0 via v1.46.0 on ap-southeast-1
❯ rustup toolchain install nightly-2020-10-25-x86_64-unknown-linux-gnu
info: syncing channel updates for 'nightly-2020-10-25-x86_64-unknown-linux-gnu'
info: latest update on 2020-10-25, rust version 1.49.0-nightly (ffa2e7ae8 2020-10-24)
info: downloading component 'cargo'
info: downloading component 'clippy'
info: downloading component 'rust-docs'
13.6 MiB / 13.6 MiB (100 %) 8.5 MiB/s in 1s ETA: 0s
info: downloading component 'rust-std'
22.3 MiB / 22.3 MiB (100 %) 8.5 MiB/s in 2s ETA: 0s
info: downloading component 'rustc'
55.1 MiB / 55.1 MiB (100 %) 6.2 MiB/s in 8s ETA: 0s
info: downloading component 'rustfmt'
info: installing component 'cargo'
info: Defaulting to 500.0 MiB unpack ram
info: installing component 'clippy'
info: installing component 'rust-docs'
info: installing component 'rust-std'
22.3 MiB / 22.3 MiB (100 %) 13.3 MiB/s in 1s ETA: 0s
info: installing component 'rustc'
55.1 MiB / 55.1 MiB (100 %) 14.2 MiB/s in 3s ETA: 0s
info: installing component 'rustfmt'
nightly-2020-10-25-x86_64-unknown-linux-gnu installed - rustc 1.49.0-nightly (ffa2e7ae8 2020-10-24)
info: checking for self-updates
example on master [?] is v0.1.0 via v1.46.0 on ap-southeast-1 took 22s
❯ rustup default nightly-2020-10-25-x86_64-unknown-linux-gnu
info: using existing install for 'nightly-2020-10-25-x86_64-unknown-linux-gnu'
info: default toolchain set to 'nightly-2020-10-25-x86_64-unknown-linux-gnu'
nightly-2020-10-25-x86_64-unknown-linux-gnu unchanged - rustc 1.49.0-nightly (ffa2e7ae8 2020-10-24)
example on master [?] is v0.1.0 via v1.49.0-nightly on ap-southeast-1
❯

Compilation

From Code to Binaries

- Formatting & Linting
- IDE Experience
- Documentation
- Compilation
- Testing
- Package Management

Compilation

From Code to Binaries

- **Formatting & Linting**
- IDE Experience
- Documentation
- Compilation
- Testing
- Package Management

rustfmt

- A tool for formatting Rust code according to style guidelines.
- Very configurable and follows the Rust style guide.
- Usually ran by running `cargo fmt` to use the multiplexing capabilities.
- Useful to enforce styling guidelines across rust repos to have common way of understanding code.

rust-clippy

- Collection of lints to catch common mistakes and find improvements.
- Over 400 lints included.
- Types:
  - Perf improvements
  - Correctness bugs
  - Idiomatic Rust code
  - Simplicity, etc

```rust
fn main() {
    let i = 0;
    while i > 5 {
        println!("inside loop");
    }
}
```

```
example on  master [?] is 📦 v0.1.0 via 🦀 v1.49.0-nightly on ☁️  ap-southeast-1
❯ cargo clippy
    Checking example v0.1.0 (/home/tarunp/work/example)
error: variables in the condition are not mutated in the loop body
 --> src/main.rs:4:11
  |
4 |     while i > 10 {
  |           ^^^^^^
  |
  = note: `#[deny(clippy::while_immutable_condition)]` on by default
  = note: this may lead to an infinite or to a never running loop
  = help: for further information visit https://rust-lang.github.io/rust-clippy/master
/index.html#while_immutable_condition

error: aborting due to previous error

error: could not compile `example`

To learn more, run the command again with --verbose.
```

Compilation

From Code to Binaries

- Formatting & Linting
- **IDE Experience**
- Documentation
- Compilation
- Testing
- Package Management

rust-analyzer

- Implementation of Language Server Protocol for Rust.
- Adds Intellisense, Refactoring, etc to your favourite Editors and IDE’s.
- Improves performance drastically compared with that of RLS.
- Leverages on-demand code analysis to be faster by performing Incremental Compilation.

Compilation

From Code to Binaries

- Formatting & Linting
- IDE Experience
- **Documentation**
- Compilation
- Testing
- Package Management

rustdoc

- Allows generation of documentation for Rust projects.
- Documentation goes hand in hand with Code i.e above the types, etc.
- Generates a markdown site that on top of the rust.docs UI framework.
- /// is syntax sugar for #[doc], which is used to write documentation.

```rust
/// A human being is represented here
pub struct Person {
    /// A person must have a name, no matter how much Juliet may hate it
    name: String,
}

impl Person {
    /// Returns a person with the name given them
    ///
    /// # Arguments
    ///
    /// * `name` - A string slice that holds the name of the person
    ///
    /// # Examples
    ///
    /// ```
    /// // You can have rust code between fences inside the comments
    /// // If you pass --test to `rustdoc`, it will even test it for you!
    /// use doc::Person;
    /// let person = Person::new("name");
    /// ```
    pub fn new(name: &str) -> Person {
        Person {
            name: name.to_string(),
        }
    }

    /// Gives a friendly hello!
    ///
    /// Says "Hello, [name]" to the `Person` it is called on.
    pub fn hello(& self) {
        println!("Hello, {}!", self.name);
    }
}
```

Struct example::Person [-][src]

[+] Show declaration

[-] A human being is represented here

Fields

`name: String`

[-] A person must have a name, no matter how much Juliet may hate it

Implementations

[-] `impl Person` [src]

[-] `pub fn new(name: &str) -> Person` [src]

Returns a person with the name given them

Arguments

- `name` - A string slice that holds the name of the person

Examples

```
// You can have rust code between fences inside the comments
// If you pass --test to `rustdoc`, it will even test it for you!
use doc::Person;
let person = Person::new("name");
```

[-] `pub fn hello(&self)` [src]

Gives a friendly hello!

Says "Hello, [name]" to the `Person` it is called on.

Auto Trait Implementations

Compilation

From Code to Binaries

- Formatting & Linting
- IDE Experience
- Documentation
- **Compilation**
- Testing
- Package Management


example on master [?] is 📦 v0.1.0 via 🦀 v1.49.0-nightly on ☁️ ap-southeast-1
❯ cargo build
   Compiling example v0.1.0 (/home/tarunp/work/example)
    Finished dev [unoptimized + debuginfo] target(s) in 0.18s
example on master [?] is 📦 v0.1.0 via 🦀 v1.49.0-nightly on ☁️ ap-southeast-1
❯ tree ./target
./target
├── CACHEDIR.TAG
└── debug
    ├── build
    ├── deps
    │   ├── example-e1cfc7d679a24b26
    │   └── example-e1cfc7d679a24b26.d
    ├── example
    ├── example.d
    ├── examples
    └── incremental
        └── example-y4o53mv32kul
            ├── s-fsqxvxzdea-te7o0l-3ewy0s42ipkr2
            │   ├── 1df20t0wrx8kb0qa.o
            │   ├── 1h2maanulcl148bg.o
            │   ├── 2n1374q0ytz3s4gn.o
            │   ├── 2xsg8loghbd362j5.o
            │   ├── 307ui5ptd6hhevxp.o
            │   ├── 30lwu03jdwu4hn9o.o
            │   ├── 33o2f5k0dljd19ey.o
            │   ├── 4sn41suyx6wtq44y.o
            │   ├── 8a0iz1i4l69tynp.o
            │   ├── dep-graph.bin
            │   ├── query-cache.bin
            │   └── work-products.bin
            └── s-fsqxvxzdea-te7o0l.lock
7 directories, 18 files

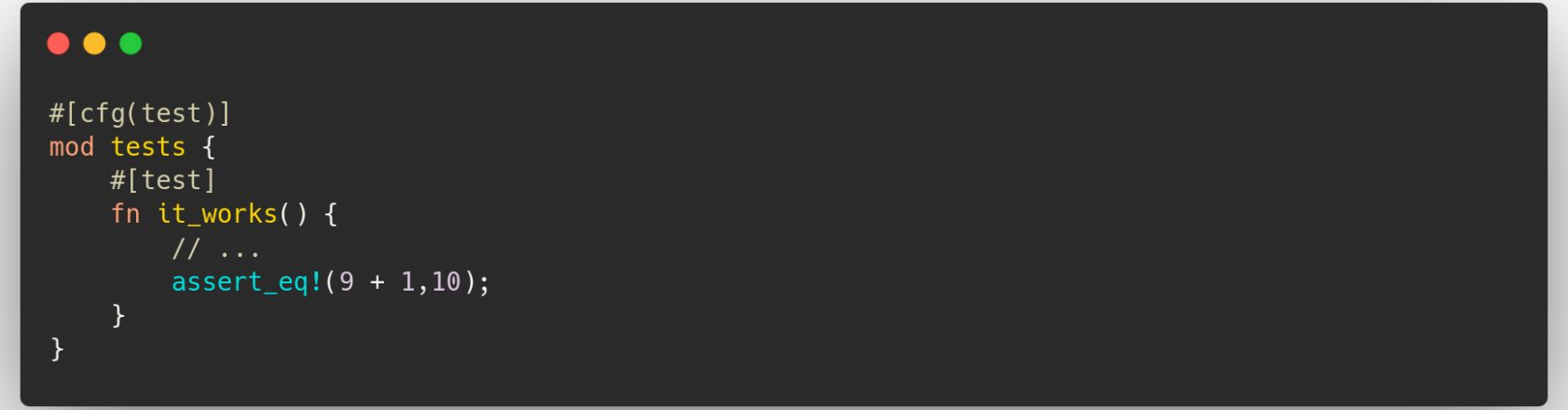
Compilation

From Code to Binaries

- Formatting & Linting
- IDE Experience
- Documentation
- Compilation
- **Testing**
- Package Management

`test` Attribute

- `cargo test` creates a test runner binary that runs functions annotated with test attribute.
- Reports are also produced on the function outcome.
- Unit tests are present in the src directory itself.
- Integration tests are present in /tests directory instead.



```rust
#[cfg(test)]
mod tests {
    #[test]
    fn it_works() {
        // ...
        assert_eq!(9 + 1,10);
    }
}
```


example on master [?] is 📦 v0.1.0 via 🦀 v1.49.0-nightly on ☁️ ap-southeast-1
❯ cargo test
warning: unused manifest key: target.armv7-unknown-linux-musleabihf.linker
   Compiling example v0.1.0 (/home/tarunp/work/example)
    Finished test [unoptimized + debuginfo] target(s) in 0.20s
     Running target/debug/deps/example-def7f933857c86de
running 1 test
test tests::example ... ok
test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

Compilation

From Code to Binaries

- Formatting & Linting
- IDE Experience
- Documentation
- Compilation
- Testing
- **Package Management**

Cargo

More than a package manager

- Dependency Management
- Workspaces
- Features
- Binary Management

Cargo

More than a package manager

- **Dependency Management**
- Workspaces
- Features
- Binary Management

Cargo

- Manage dependencies and have repeatable builds.
- Metadata files to keep track of the package information.
- Performs builds by fetching package dependencies.
- Introduces a package layout.
- Acts like an umbrella tool for most operations.

```
[package]
name = "example"
version = "0.1.0"
authors = ["Your Name <you@example.com>"]
edition = "2018"

[dependencies]
time = "0.1.12"
regex = "0.1.41"
```

Cargo

More than a package manager

- Dependency Management
- **Workspaces**
- Features
- Binary Management

Cargo Workspaces

- Workspace allows grouping a set of packages.
- Each package can be a binary or a library crate.
- It can help manage multiple related packages.
- Configured by adding a [workspace] section into Cargo.toml
- Packages share a common Cargo.lock and output directories (i.e target).

```
[workspace]

members = [
    "tracing",
    "tracing-core",
    "tracing-attributes",
    "tracing-error",
    "tracing-flame",
    "tracing-futures",
    "tracing-tower",
    "tracing-log",
    "tracing-macros",
    "tracing-opentelemetry",
    "tracing-subscriber",
    "tracing-serde",
    "tracing-appender",
    "tracing-journald",
    "examples"
]
```

Cargo

More than a package manager

- Dependency Management
- Workspaces
- **Features**
- Binary Management

Features

- Rust compiler has built in support for compile time feature flags.
- Based on the feature flags configuration, The compilation is affected.
- This is possible by using the `cfg` attributes in code.
- Very useful for packages to have multiple feature levels based on the dependencies.
- A feature of a package is either an optional dependency, or a set of other features.

```
[features]
default = ["std"]
alloc = []
std = ["lazy_static", "alloc"]

[badges]
maintenance = { status = "actively-developed" }

[dependencies]
lazy_static = { version = "1", optional = true }
```

```rust
#[cfg(feature = "std")]
mod inner {
    // implementation with the usage of `std`
    // ...
}


#[cfg(not(feature = "std"))]
mod inner {
    // implementation without the usage of `std`
    // ...
}
```


[dependencies]
tracing-core = { version = "0.2", default-features = false, features = ["alloc"] }

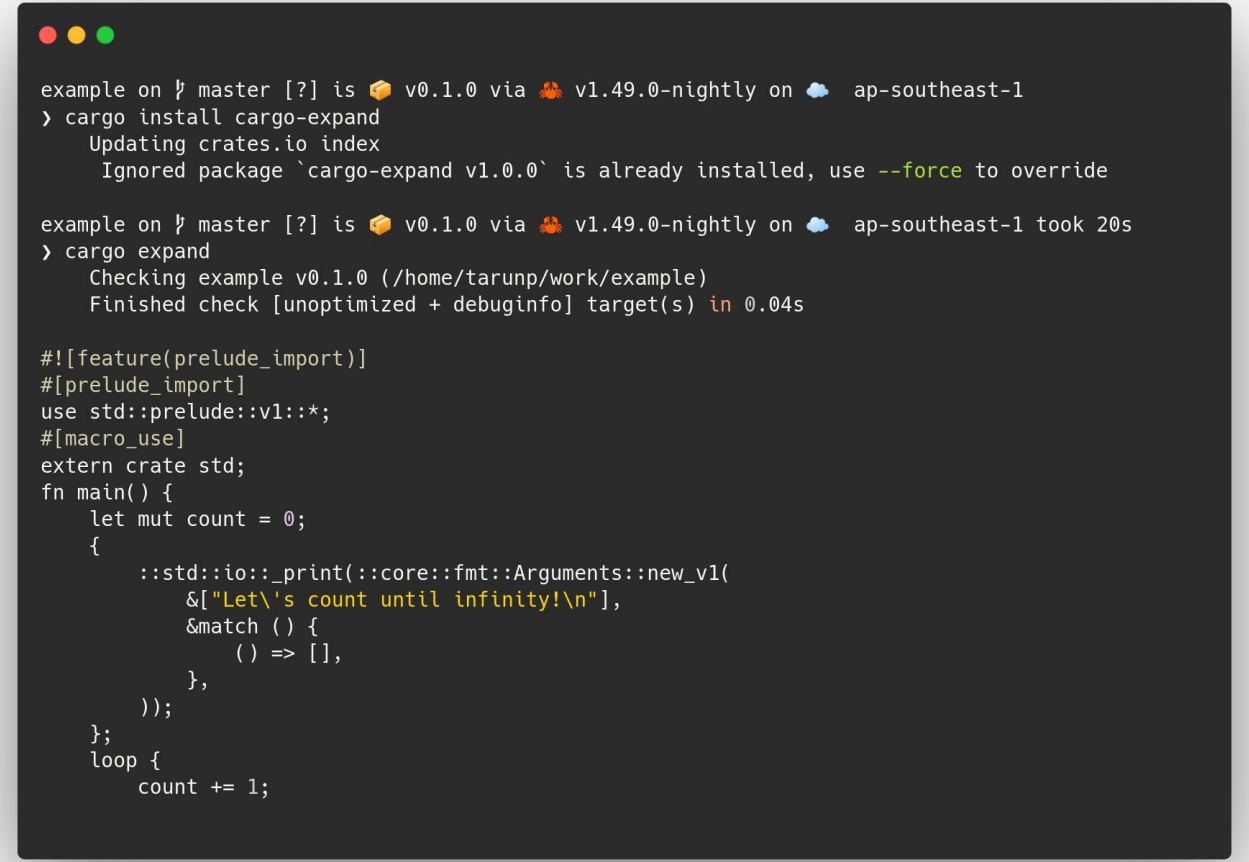
Cargo

More than a package manager

- Dependency Management
- Workspaces
- Features
- **Binary Management**

Cargo with Binaries

- Cargo is built to be extensible with new commands without having to modify cargo itself.
- `cargo expand` invokes `cargo-expand` from $PATH.
- Binaries can also be published on crates.io
- Cargo install can be used to retrieve and install binaries.
- These binaries are installed into `$HOME/.cargo` unless overridden.



```
example on  master [?] is 📦 v0.1.0 via 🦀 v1.49.0-nightly on ☁️  ap-southeast-1
❯ cargo install cargo-expand
    Updating crates.io index
     Ignored package `cargo-expand v1.0.0` is already installed, use --force to override

example on  master [?] is 📦 v0.1.0 via 🦀 v1.49.0-nightly on ☁️  ap-southeast-1 took 20s
❯ cargo expand
    Checking example v0.1.0 (/home/tarunp/work/example)
    Finished check [unoptimized + debuginfo] target(s) in 0.04s

#![feature(prelude_import)]
#[prelude_import]
use std::prelude::v1::*;
#[macro_use]
extern crate std;
fn main() {
    let mut count = 0;
    {
        ::std::io::_print(::core::fmt::Arguments::new_v1(
            &["Let\'s count until infinity!\n"],
            &match () {
                () => [],
            },
        ));
    };
    loop {
        count += 1;
```

Debugging

Finding bugs and runtime diagnostics.

- Logging
- Tracing
- GDB

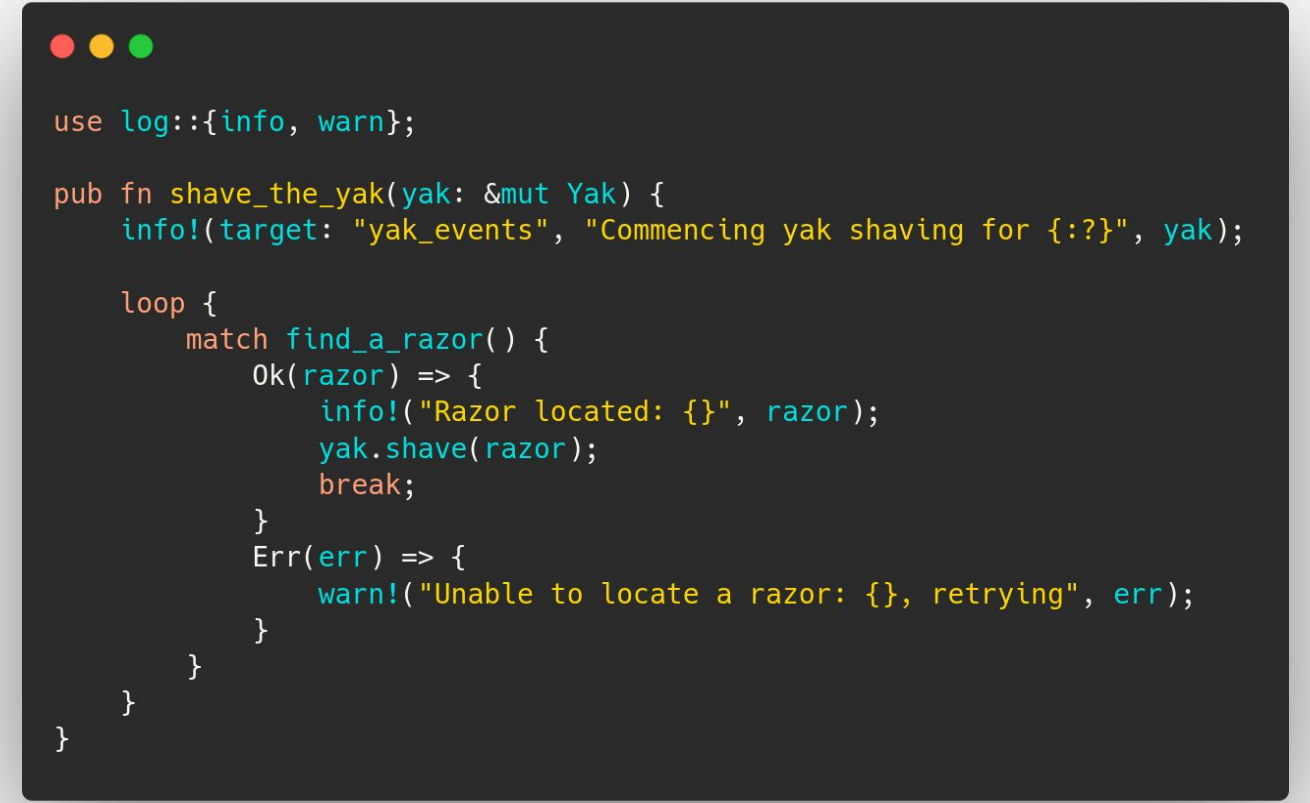
Debugging

Finding bugs and runtime diagnostics.

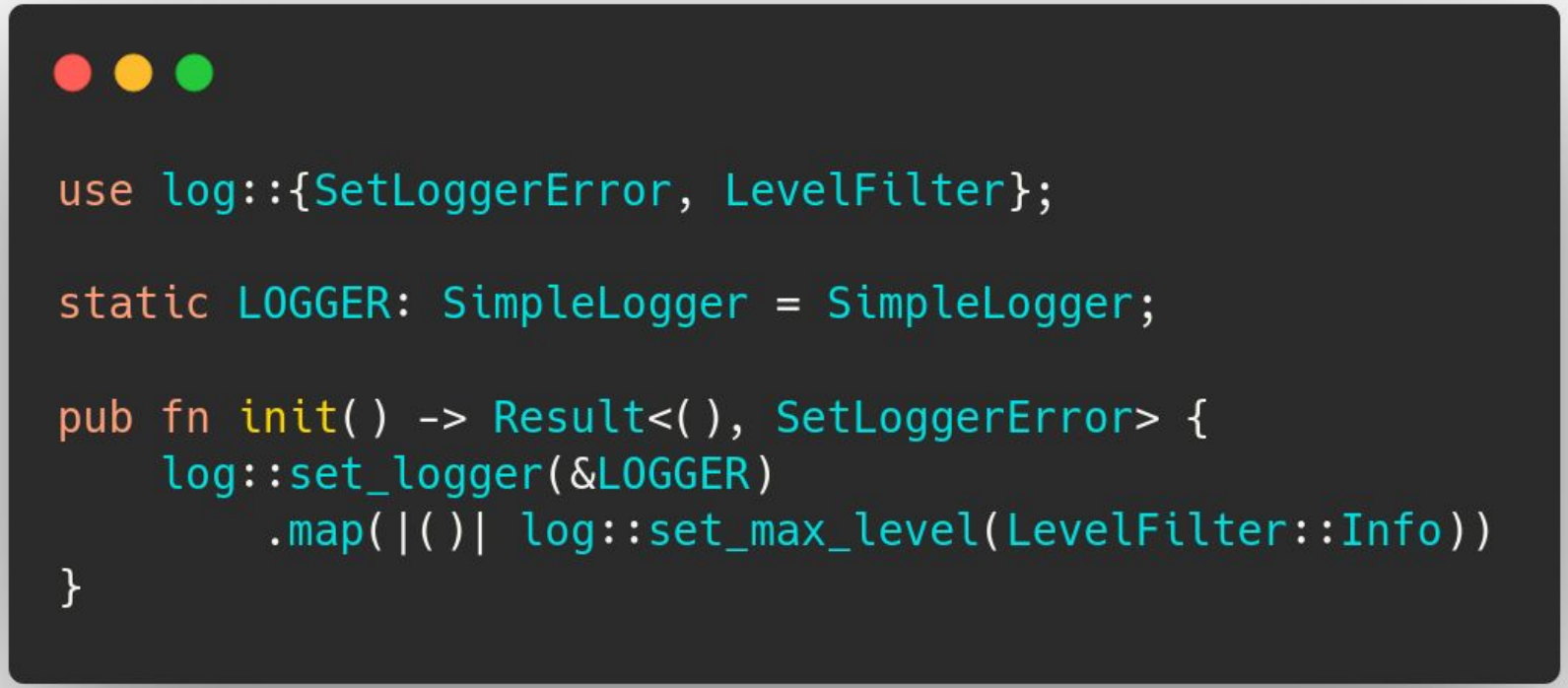
- **Logging**
- Tracing
- GDB

Crate log

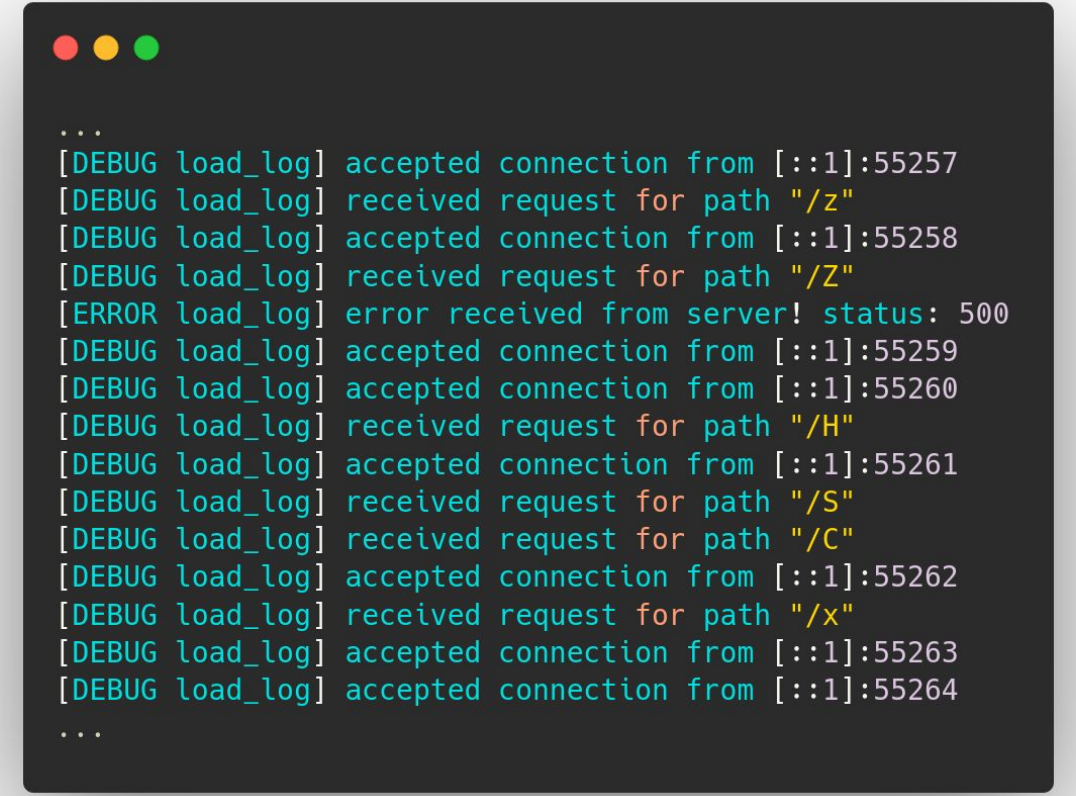
- Contains `debug`, `error`, `info`, `log`, `trace`, `warn` macros to report.
- Abstracts over the actual logging implementation.
- Consumer of a library can decide which implementation they want to use.
- Low overhead when no implementation is specified.
- Simple API to implement your own logger implementation



```rust
use log::{info, warn};

pub fn shave_the_yak(yak: &mut Yak) {
    info!(target: "yak_events", "Commencing yak shaving for {:?}", yak);

    loop {
        match find_a_razor() {
            Ok(razor) => {
                info!("Razor located: {}", razor);
                yak.shave(razor);
                break;
            }
            Err(err) => {
                warn!("Unable to locate a razor: {}, retrying", err);
            }
        }
    }
}
```



```rust
use log::{SetLoggerError, LevelFilter};

static LOGGER: SimpleLogger = SimpleLogger;

pub fn init() -> Result<(), SetLoggerError> {
    log::set_logger(&LOGGER)
        .map(|()| log::set_max_level(LevelFilter::Info))
}
```


...
[DEBUG load_log] accepted connection from [::1]:55257
[DEBUG load_log] received request for path "/z"
[DEBUG load_log] accepted connection from [::1]:55258
[DEBUG load_log] received request for path "/Z"
[ERROR load_log] error received from server! status: 500
[DEBUG load_log] accepted connection from [::1]:55259
[DEBUG load_log] accepted connection from [::1]:55260
[DEBUG load_log] received request for path "/H"
[DEBUG load_log] accepted connection from [::1]:55261
[DEBUG load_log] received request for path "/S"
[DEBUG load_log] received request for path "/C"
[DEBUG load_log] accepted connection from [::1]:55262
[DEBUG load_log] received request for path "/x"
[DEBUG load_log] accepted connection from [::1]:55263
[DEBUG load_log] accepted connection from [::1]:55264
...

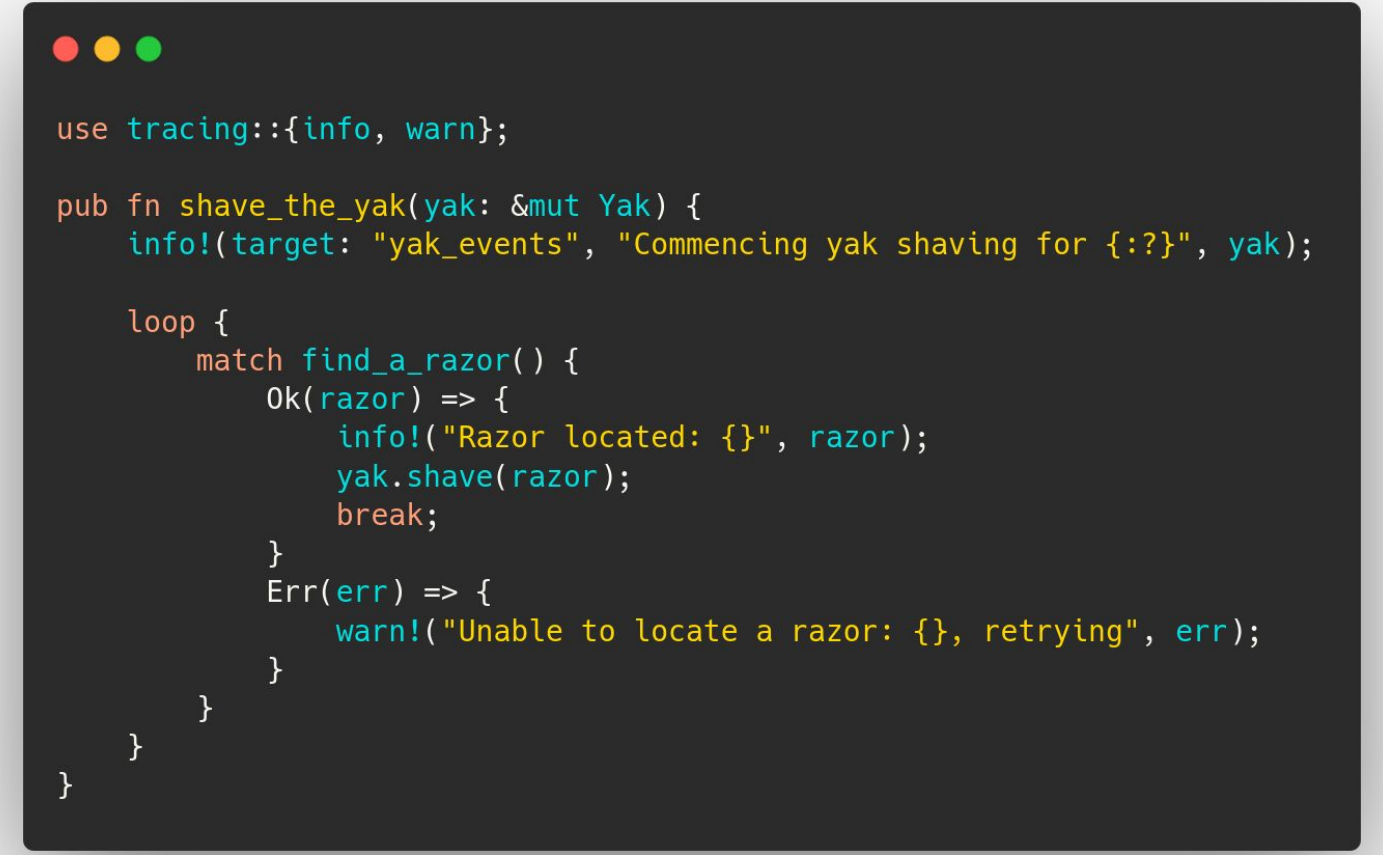
Debugging

Finding bugs and runtime diagnostics.

- Logging
- **Tracing**
- GDB

Crate tracing

- More than a logging library.
- Same simple API for consumers.
- Implements scoped, contextual, and structured diagnostic instrumentation.
- Introduces a new primitive called Span, which represents a period of time.
- Useful for Asynchronous Systems, Distributed Tracing Instrumentation, etc.



```rust
use tracing::{info, warn};

pub fn shave_the_yak(yak: &mut Yak) {
    info!(target: "yak_events", "Commencing yak shaving for {:?}", yak);

    loop {
        match find_a_razor() {
            Ok(razor) => {
                info!("Razor located: {}", razor);
                yak.shave(razor);
                break;
            }
            Err(err) => {
                warn!("Unable to locate a razor: {}, retrying", err);
            }
        }
    }
}
```


use tracing::instrument;
#[instrument]
pub async fn connect_to(remote: SocketAddr) -> io::Result<TcpStream> {
// ...
trace!(bytes_read, messages = num_processed);
}


TRACE request{req.method=GET req.path="/z"}: load: handling request...
TRACE request{req.method=GET req.path="/z"}: load: error=i don't like this letter. letter="z"
TRACE request{req.method=GET req.path="/z"}: load: rsp.error=unknown internal error
ERROR load_gen{remote.addr=[::1]:3000}:request{req.method=GET req.path="/z"}: gen: error received from server! status=500
TRACE load_gen{remote.addr=[::1]:3000}:request{req.method=GET req.path="/z"}: gen: response complete. rsp.body=unknown internal error
TRACE load_gen{remote.addr=[::1]:3000}:request{req.method=GET req.path="/z"}: gen: sending request...
TRACE load_gen{remote.addr=[::1]:3000}:request{req.method=GET req.path="/z"}: tower_buffer::service: sending request to buffer worker
DEBUG request{req.method=GET req.path="/z"}: load: received request. req.headers={"content-length": "24", "host": "[::1]:3000"} req.version=HTTP/1.1
TRACE request{req.method=GET req.path="/z"}: load: handling request...
TRACE request{req.method=GET req.path="/z"}: load: error=i don't like this letter. letter="z"
TRACE request{req.method=GET req.path="/z"}: load: rsp.error=unknown internal error

Debugging

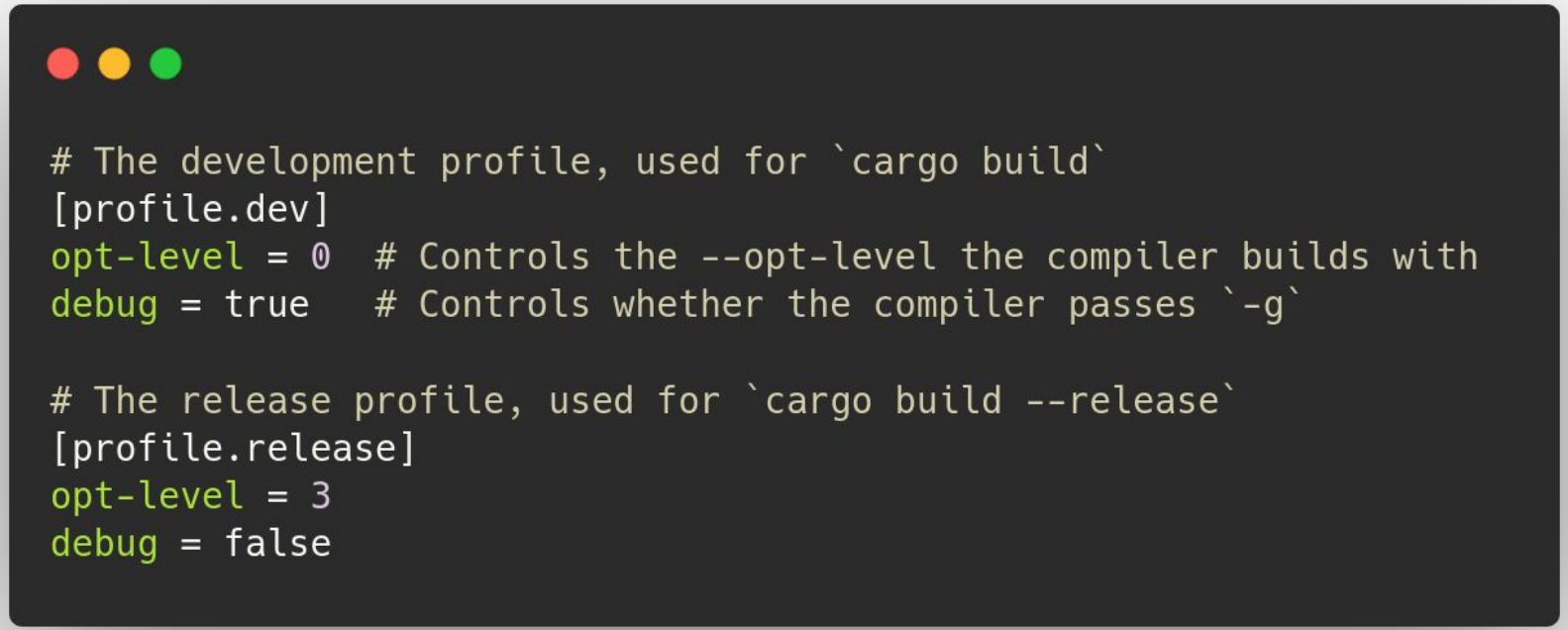
Finding bugs and runtime diagnostics.

- Logging
- Tracing
- **GDB**

GDB (GNU Debugger)

- GNU Project debugger, allows us to understand what is going on inside the program while it executes.
- Using GDB, the program's running can be controlled and get information from inside the code.
- It allows users to apply breakpoints and retrieve runtime information i.e variables, stacks, etc.
- It also has support for various languages like C, C++, Go, etc.
- `rust-gdb` is a wrapper that provides pretty printers specific to rust, etc.

```rust
fn main() {
    let mut count = 0u32;
    println!("Let's count until infinity!");
    // Infinite loop
    loop {
        count += 1;
        if count == 3 {
            println!("three");

            // Skip the rest of this iteration
            continue;
        }
        println!("{}", count);
        if count == 5 {
            println!("OK, that's enough");

            // Exit this loop
            break;
        }
    }
}
```



```
# The development profile, used for `cargo build`
[profile.dev]
opt-level = 0  # Controls the --opt-level the compiler builds with
debug = true   # Controls whether the compiler passes `-g`

# The release profile, used for `cargo build --release`
[profile.release]
opt-level = 3
debug = false
```


example on master [?] is v0.1.0 via v1.49.0-nightly on ap-southeast-1 took 34s
❯ gdb -q ./target/debug/example
Reading symbols from ./target/debug/example...
warning: Missing auto-load script at offset 0 in section .debug_gdb_scripts
of file /home/tarunp/work/example/target/debug/example.
Use `info auto-load python-scripts [REGEXP]' to list them.
(gdb) b 8
Breakpoint 1 at 0x53b2: file src/main.rs, line 8.
(gdb) r
Starting program: /home/tarunp/work/example/target/debug/example
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".
Let's count until infinity!
1
2
Breakpoint 1, example::main () at src/main.rs:8
8   println!("three");
(gdb) p count
$1 = 3
(gdb)

Thank You! 
Questions?