# Roadmap 2021

# Summary

This Roadmap lays out the plans for Clippy in 2021:

- Improving usability and reliability
- Improving experience of contributors and maintainers
- Develop and specify processes

Members of the Clippy team will be assigned tasks from one or more of these topics. The team member is then responsible to complete the assigned tasks. This can either be done by implementing them or by providing mentorship to interested contributors.

# Motivation

With the ongoing growth of the Rust language and with that of the whole ecosystem, also Clippy gets more and more users and contributors. This is good for the project, but also brings challenges along. Some of these challenges are:

- More issues about reliability or usability are popping up
- Traffic is hard to handle for a small team
- Bigger projects don't get completed due to the lack of processes and/or time of the team members

Additionally, according to the Rust Roadmap 2021, clear processes should be defined by every team and unified across teams. This Roadmap is the first step towards this.

# Explanation

This section will explain the things that should be done in 2021. It is important to note, that this document focuses on the "What?", not the "How?". The later will be addressed in follow-up tracking issue, with an assigned team member.

The following is split up in two major sections. The first section covers the user facing plans, the second section the internal plans.

## User Facing

Clippy should be as pleasant to use and configure as possible. This section covers plans that should be implemented to improve the situation of Clippy in this regard.

### Usability

In the following, plans to improve the usability are covered.

#### No Output After `cargo check`

Currently when `cargo clippy` is run after `cargo check`, it does not produce any output. This is especially problematic since `rust-analyzer` is on the rise and it uses `cargo check` for checking code. A fix is already implemented, but it still has to be pushed over the finish line. This also includes the stabilization of the `cargo clippy --fix` command or the support of multi-span suggestions in `rustfix`.

- #4612

#### `lints.toml` Configuration

This is something that comes up every now and then: a reusable configuration file, where lint levels can be defined. Discussions about this often lead to nothing specific or to "we need an RFC for this". And this is exactly what needs to be done. Get together with the cargo team and write an RFC and implement such a configuration file somehow and somewhere.

- #3164
- cargo#5034
- IRLO

#### Lint Groups

There are more and more issues about managing lints in Clippy popping up. Lints are hard to implement with a guarantee of no/few false positives (FPs). One way to address this might be to introduce more lint groups to give users the ability to better manage lints, or improve the process of classifying lints, so that disabling lints due to FPs becomes rare. It is important to note, that Clippy lints are less conservative than `rustc` lints, which won't change in the future.

- #5537
- #6366

### Reliability

In the following, plans to improve the reliability are covered.

#### False Positive Rate

In the worst case, new lints are only available in nightly for 2 weeks, before hitting beta and ultimately stable. This and the fact that fewer people use nightly Rust nowadays makes it more probable that a lint with many FPs hits stable. This leads to annoyed users, that will disable these new lints in the best case and to more annoyed users, that will stop using Clippy in the worst. A process should be developed and implemented to prevent this from happening.

- #6429

## Internal

(The end of) 2020 has shown, that Clippy has to think about the available resources, especially regarding management and maintenance of the project. This section address issues affecting team members and contributors.

### Management

In 2020 Clippy achieved over 1000 open issues with regularly between 25-35 open PRs. This is simultaneously a win and a loss. More issues and PRs means more people are interested in Clippy and in contributing to it. On the other hand, it means for team members more work and for contributors longer wait times for reviews. The following will describe plans how to improve the situation for both team members and contributors.

#### Clear Expectations for Team Members

According to the Rust Roadmap 2021, a document specifying what it means to be a member of the team should be produced. This should not put more pressure on the team members, but rather help them and interested folks to know what the expectations are. With this it should also be easier to recruit new team members and may encourage people to get in touch, if they're interested to join.

#### Scaling up the Team

More people means less work for each individual. Together with the document about expectations for team members, a document defining the process of how to join the team should be produced. This can also increase the stability of the team, in case of current members dropping out (temporarily). There can also be different roles in the team, like people triaging vs. people reviewing.

#### Regular Meetings

Other teams have regular meetings. Clippy is big enough that it might be worth to also do them. Especially if more people join the team, this can be important for sync-ups. Besides the asynchronous communication, that works well for working on separate lints, a meeting adds a synchronous alternative at a known time. This is especially helpful if there are bigger things that need to be discussed (like the projects in this roadmap). For starters bi-weekly meetings before Rust syncs might make sense.

#### Triaging

To get a handle on the influx of open issues, a process for triaging issues and PRs should be developed. Officially, Clippy follows the Rust triage process, but currently no one enforces it. This can be improved by sharing triage teams across projects or by implementing dashboards / tools which simplify triaging.

### Development

Improving the developer and contributor experience is something the Clippy team works on regularly. Though, some things might need special attention and planing. These topics are listed in the following.

#### Process for New and Existing Lints

As already mentioned above, classifying new lints gets quite hard, because the probability of a buggy lint getting into stable is quite high. A process should be implemented on how to classify lints. In addition, a test system should be developed to find out which lints are currently problematic in real world code to fix or disable them.

- #6429 (comment)
- #6429 (comment)

#### Processes

Related to the point before, a process for suggesting and discussing major changes should be implemented. It's also not clearly defined when a lint should be enabled or disabled by default. This can also be improved by the test system mentioned above.

#### Dev-Tools

There's already `cargo dev` which makes Clippy development easier and more pleasant. This can still be expanded, so that it covers more areas of the development process.

- #5394

#### Contributor Guide

Similar to a Clippy Book, which describes how to use Clippy, a book about how to contribute to Clippy might be helpful for new and existing contributors. There's already the `doc` directory in the Clippy repo, this can be turned into a `mdbook`.

#### `rustc` integration

Recently Clippy was integrated with `git subtree` into the `rust-lang/rust` repository. This made syncing between the two repositories easier. A `#[non_exhaustive]` list of things that still can be improved is:

1. Use the same `rustfmt` version and configuration as `rustc`.
2. Make `cargo dev` work in the Rust repo, just as it works in the Clippy repo. E.g. `cargo dev bless` or `cargo dev update_lints`. And even add more things to it that might be useful for the Rust repo, e.g. `cargo dev deprecate`.
3. Easier sync process. The `subtree` situation is not ideal.

### Prioritization

The most pressing issues for users of Clippy are of course the user facing issues. So there should be a priority on those issues, but without losing track of the internal issues listed in this document.

Getting the FP rate of warn/deny-by-default lints under control should have the highest priority. Other user facing issues should also get a high priority, but shouldn't be in the way of addressing internal issues.

To better manage the upcoming projects, the basic internal processes, like meetings, tracking issues and documentation, should be established as soon as possible. They might even be necessary to properly manage the projects, regarding the user facing issues.

## Prior Art

### Rust Roadmap

Rust's roadmap process was established by RFC 1728 in 2016. Since then every year a roadmap was published, that defined the bigger plans for the coming years. This years roadmap can be found here.

## Drawbacks

### Big Roadmap

This roadmap is pretty big and not all items listed in this document might be addressed during 2021. Because this is the first roadmap for Clippy, having open tasks at the end of 2021 is fine, but they should be revisited in the 2022 roadmap.