



# Turning Git commits into changelog with `git-cliff`

Orhun Parmaksız

# whoami

- Open Source Developer (@orhun on GitHub)
  - Maintainer @ Ratatui.rs
  - Creator @ {git-cliff,kmon,gpg-tui,systemoid,daktilo,...}
- Package Maintainer @ Arch Linux / Alpine Linux
- Paid Open Source Contributor @ Shuttle.rs



 **git-cliff** Public



A highly customizable Changelog Generator that follows Conventional Commit specifications 🏔

 Rust  6.2k  119

 **kmon** Public



Linux Kernel Manager and Activity Monitor 🐧🇺🇸

 Rust  2.2k  65

 **gpg-tui** Public



Manage your GnuPG keys with ease! 🗝

 Rust  1.1k  32

 **systemoid** Public



A more powerful alternative to sysctl(8) with a terminal user interface 🐧

 Rust  1.1k  19

 **daktilo** Public



Turn your keyboard into a typewriter! 🖨

 Rust  749  17


 **linuxwave** Public



Generate music from the entropy of Linux 🐧🎵

 Zig  429  11

# schedule()

- Problem description
- Possible solutions
- The story of `git-cliff`
- Demo implementation 

Imagine a developer

# Imagine a developer

- uses git

# Imagine a developer

- uses git
  - uses conventional commits

# Imagine a developer

- uses git
  - uses conventional commits
    - <type>[optional scope]: <description>*
    - [optional body]*
    - [optional footer(s)]*



# Imagine a developer

- uses git
  - uses conventional commits
    - feat(life)!: set the computer on fire*

# Imagine a developer

- uses git
  - uses conventional commits

*<https://www.conventionalcommits.org/en/v1.0.0/>*

# Imagine a developer

- uses git
  - uses conventional commits
- uses semantic versioning

# Imagine a developer

- uses git
  - uses conventional commits
- uses semantic versioning

*MAJOR.MINOR.PATCH*

# Imagine a developer

- uses git
  - uses conventional commits
- uses semantic versioning

*<https://semver.org/>*

# Imagine a developer

- uses git
  - uses conventional commits
- uses semantic versioning
- likes automation

# Imagine a developer

- uses git
  - uses conventional commits
- uses semantic versioning
- likes automation
- wants to create a release

Changelog? 🤔



A changelog is a log or record of all **notable changes** made to a project. The project is often a website or software project, and the changelog usually includes records of changes such as **bug fixes**, **new features**, etc. Some open-source projects include a changelog as one of the top-level files in their distribution.

<https://en.wikipedia.org/wiki/Changelog>

May 1, 2018

 BurntSushi

 1.0.0 

 b5ef0ec 

Compare 

# 1.0.0 Latest

This release marks the 1.0 release of regex.

While this release includes some breaking changes, most users of older versions of the regex library should be able to migrate to 1.0 by simply bumping the version number. The important changes are as follows:

- We adopt Rust 1.20 as the new minimum supported version of Rust for regex. We also tentatively adopt a policy that permits bumping the minimum supported version of Rust in minor version releases of regex, but no patch releases. That is, with respect to semver, we do not strictly consider bumping the minimum version of Rust to be a breaking change, but adopt a conservative stance as a compromise.
- Octal syntax in regular expressions has been disabled by default. This permits better error messages that inform users that backreferences aren't available. Octal syntax can be re-enabled via the corresponding option on `RegexBuilder`.

# Changelog

---

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](#), and this project adheres to [Semantic Versioning](#).

## [1.7.0] - 2023-03-01

---

### Added

- Add WebP encoding support
- Use LZW-compressed frames for the default GIF encoder

### Changed

- Update sponsorship options
- Optimize Dockerfile
- Bump dependencies

### Removed

- Remove deprecated compression options from PNG encoder

Automation 🤖

v0.24.1

Target: main

Excellent! This tag will be created from the target when you publish this release.

v0.24.1

Write

Preview

H B I       @  

Generate release notes

## ## What's Changed

- \* fix: MSRV is now `1.70.0` by @a-kenji in <https://github.com/ratatui-org/ratatui/pull/593>
- \* docs(examples): fix run `demo2` command error by @rikonaka in <https://github.com/ratatui-org/ratatui/pull/595>
- \* ci(codecov): adjust threshold and noise settings by @joshka in <https://github.com/ratatui-org/ratatui/pull/615>

## ## New Contributors

- \* @rikonaka made their first contribution in <https://github.com/ratatui-org/ratatui/pull/595>

\*\*Full Changelog\*\*[: https://github.com/ratatui-org/ratatui/compare/v0.24.0...v0.24.1](https://github.com/ratatui-org/ratatui/compare/v0.24.0...v0.24.1)

# git shortlog

24seconds (1):

Add rust-sadari-cli in `Apps using tui` section (#278)

Aatu Kajasto (1):

fix(chart): use graph style for top line (#462)

Aizon (2):

docs(table): add documentation for `Table::new()` (#471)

docs(table): add documentation for `block` and `header` methods of the `Table` widget (#505)

Alexander Batischev (1):

feat(terminal): Add after-draw() cursor control to Frame (#91) (#309)

Alexandru Scvortov (2):

fix: actually clear buffer in TestBackend::clear (#461)

# Other tools

- <https://github.com/saschagrunert/git-journal>
- <https://github.com/clog-tool/clog-cli>
- <https://crates.io/crates/relnotes>
- <https://github.com/oknozor/cocoqitto>
- <https://github.com/rustic-games/jilu>
- <https://github.com/github-changelog-generator/github-changelog-generator>

# The problem

Creating a changelog is a tedious task.

It can be easily automated with the help of **conventional commits** and **semantic versioning**.



```
* df6aef4 (HEAD -> master, origin/master, origin/HEAD) feat(cache): use cache while fetching pages
* a9d4050 feat(config): support multiple file formats
* 06412ac (tag: v1.0.1) chore(release): add release script
* e4fd3cf refactor(parser): expose string functions
* ad27b43 (tag: v1.0.0) docs(example)!: add tested usage example
* 9add0d4 fix(args): rename help argument due to conflict
* a140cef feat(parser): add ability to parse arrays
* 81fbc63 docs(project): add README.md
* a78bc36 Initial commit
```

```
* df6aef4 (HEAD -> master, origin/master, origin/HEAD) feat(cache): use cache while fetching pages
* a9d4050 feat(config): support multiple file formats
* 06412ac (tag: v1.0.1) chore(release): add release script
* e4fd3cf refactor(parser): expose string functions
* ad27b43 (tag: v1.0.0) docs(example)!: add tested usage example
* 9add0d4 fix(args): rename help argument due to conflict
* a140cef feat(parser): add ability to parse arrays
* 81fbc63 docs(project): add README.md
* a78bc36 Initial commit
```



```
* df6aef4 (HEAD -> master, origin/master, origin/HEAD) feat(cache): use cache while fetching pages
* a9d4050 feat(config): support multiple file formats
* 06412ac (tag: v1.0.1) chore(release): add release script
* e4fd3cf refactor(parser): expose string functions
* ad27b43 (tag: v1.0.0) docs(example)!: add tested usage example
* 9add0d4 fix(args): rename help argument due to conflict
* a140cef feat(parser): add ability to parse arrays
* 81fbc63 docs(project): add README.md
* a78bc36 Initial commit
```



```
# Changelog
All notable changes to this project will be documented in this file.

## [unreleased]
### 🏔 Features
- *(cache)* Use cache while fetching pages
- *(config)* Support multiple file formats
-
## [1.0.1] - 2021-07-18
### 🛠 Refactor
- *(parser)* Expose string functions
### ⚙ Miscellaneous Tasks
- *(release)* Add release script
```



**CHANGELOG IS  
JUST A BEAUTIFIED  
GIT HISTORY**

**ALWAYS  
HAS BEEN**

(not always)



git-cliff

Imagine a developer

# Imagine a developer

- writes Rust 



# Imagine a developer

- writes Rust 🦀
- created a cool project 😎

# Imagine a developer

- writes Rust 🦀
- created a cool project 😎
- **wants to create a changelog** 🤔

This was me.



<https://github.com/orhun/gpg-tui>

# GPCG TUI

Manage your GnuPG keys with ease!

# jilu

- generates a changelog based on the state of your Git repository.
- convert conventional commits into a human readable changelog
- use Git tags to annotate your releases with release titles and richly formatted release notes
- customize your changelog template to best serve your community
- integrate the jilu binary into your CI workflow for automated updates

记录

<https://github.com/rustic-games/jilu>

## Changes

### Features

- **allow LineGauge background styles** ([ ff3a293 ])

LineGauge can be styled with regard to the line itself.

There was no option to style the background behind the line.

This commit adds this feature.

Implements ratatui-org#424

- **implement common traits for WindowSize (#586)** ([ c6c3f88 ])

# TEMPLATE.md

## Changelog [↗](#)

---

All notable changes to this project are documented in this file.

The format is based on [Keep a Changelog](#), and this project adheres to [Semantic Versioning](#). The file is auto-generated using [Conventional Commits](#).

## Overview [↗](#)

---

- [unreleased](#)

{%- for release in releases %}

- [ `{{ release.version }}` ](`#{{ release.version | replace(from=".", to="" )}}`) – `{{ release.date | date(format="%Y.%m.%d")}}` {%- endfor %}

## **[Unreleased]** [↗](#)

---

{% if unreleased.changes -%} {%- for change in unreleased.changes -%}

- `{{ change.type }}`: `{{ change.description }}` [ `{{ change.commit.short_id }}` ] {%- endfor %} {%- else -%} *nothing new to show for... yet!*

{% endif -%} {%- for release in releases -%}

## **[`{{ release.version }}`]** {%- if release.title %} – **`{{ release.title }}`** {%- endif %} [↗](#)

---

`{{ release.date | date(format="%Y.%m.%d")}}` {%- if release.notes %}

`{{ release.notes }}` {%- endif -%} {%- if release.changeset.contributors %}





**jcgruenhage** and **JeanMertz** chore: update convention...

c334c5d on Aug 17, 2020

49 commits

src	feat: provide scopeheader filter	3 years ago
.gitignore	feat: initial working version	4 years ago
CHANGELOG.md	chore: update change log	3 years ago
Cargo.lock	chore: prepare to release v0.4.0	3 years ago
Cargo.toml	chore: prepare to release v0.4.0	3 years ago
README.md	chore: prepare to release v0.4.0	3 years ago
rustfmt.toml	feat: initial working version	4 years ago
template.md	chore: update conventional commit link in template	3 years ago

**README.md**

Generate a change log based on the state of your Git repository.

[Readme](#)

[Activity](#)

**103** stars

**4** watching

**6** forks

**4** years old

[Report repository](#)

**Releases** 5

**Commit scope templating** Latest

on Aug 17, 2020

---

May 8, 2021

12:29 AM **orhun** I ditched that idea btw

**orhun** deps.rs already shows audit output

**orhun** so I am out of ideas for now

**orhun** but I'll drink lemon juice mixed sprite, eat couple of caster sugar-dipped fresh strawberries and come back with ideas soon

3:19 PM **orhun** let's write a decent changelog generator

**orhun** I can't find any good ones

**\*SLURP\***



**AND THAT'S HOW IT ALL STARTED..**

makeameme.org

# What do we need to do?

1. Open Git repository (read contents)
2. Parse commits
3. Generate changelog
4. Profit

12:12 AM orhun

```
use chrono::Utc;
use git2::{Commit, Repository, Sort};
use std::env;
use std::io::{self, Write};

fn main() {
    // Init repository
    let repo = Repository::open(env::var("REPOSITORY").unwrap()).unwrap();
    let mut revwalk = repo.revwalk().unwrap();
    revwalk.set_sorting(Sort::NONE | Sort::TIME).unwrap();
    revwalk.push_head().unwrap();

    // Parse commits
    let mut fixes = Vec::new();
    let mut features = Vec::new();
    for commit in revwalk
        .filter_map(|id| id.ok())
        .filter_map(|id| repo.find_commit(id).ok())
    {
        let message = commit.message().unwrap();
        if message.starts_with("feat") {
            features.push(commit);
        } else if message.starts_with("fix") {
            fixes.push(commit);
        }
    }

    // Generate changelog
    let generate = |commits: Vec<Commit>| -> Vec<String> {
        commits
            .iter()
            .map(|commit| {
                let id = commit.id().to_string();
                format!(
                    "* {} ({})({})",
                    commit.message().unwrap().trim(),
                    &id[0..7],
                    &id[0..7],
                )
            })
            .collect()
    };
};
```

Step 1

# Reading Git contents

## git2 v0.18.1

1

Bindings to libgit2 for interoperating with git repositories. This library is both threadsafe both reading and writing git repositories.

[#git](#)

[Readme](#)

[116 Versions](#)

[Dependencies](#)

[Dependents](#)

## git2-rs

[Documentation](#)

libgit2 bindings for Rust.

[\[dependencies\]](#)

```
git2 = "0.18.1"
```

## gitoxide v0.31.1

2

A command-line application for interacting with git repositories

[Readme](#)

[38 Versions](#)

[Dependencies](#)

[Dependents](#)

Rust [crates.io v0.31.1](#) [rustc 1.65.0+](#)

gitoxide is an implementation of `git` written in Rust for developing future-proof applications which strive for correctness and performance while providing a pleasant and unsurprising developer experience.

```
Command::new("git")  
    .args(["log", "--pretty=format:%H", "-n", "1"])  
    .spawn()  
    .unwrap()
```

3

git2



# git2

- requires libgit2 >=1.7.1

# git2

- requires libgit2 >=1.7.1
- vendored-libgit2 to always compile and statically link to a copy of libgit2

# git2

- requires libgit2 >=1.7.1
- vendored-libgit2 to always compile and statically link to a copy of libgit2
- LIBGIT2\_NO\_VENDOR=1

# Cargo.toml

```
[dependencies]  
git2 = "0.18.1"
```

```
1 // Initialize repository
2 let path = std::env::var("REPOSITORY"?);
3 let repo = git2::Repository::open(path)?;
4
5 let mut revwalk = repo.rewalk()?;
6 revwalk.set_sorting(git2::Sort::NONE | git2::Sort::TIME)?;
7
8 revwalk.push_head()?;
```

Step 2

```
1 // Parse commits
2 let mut fixes = Vec::new();
3 let mut features = Vec::new();
4
5 for commit in revwalk
6     .filter_map(|id| id.ok())
7     .filter_map(|id| repo.find_commit(id).ok())
8 {
9     let message = commit.message().unwrap();
10
11     if message.starts_with("feat") {
12         features.push(commit);
13     } else if message.starts_with("fix") {
14         fixes.push(commit);
15     }
16
17 }
```

Fixes:

```
[
  Commit {
    id: 9add0d4616dc95a6ea8b01d5e4d233876b6e5e00,
    summary: "fix(args): rename help argument due to conflict",
  },
]
```

Features:

```
[
  Commit {
    id: df6aef41292f3ffe5887754232e6ea7831c50ba5,
    summary: "feat(cache): use cache while fetching pages",
  },
  Commit {
    id: a9d4050212a18f6b3bd76e2e41fbb9045d268b80,
    summary: "feat(config): support multiple file formats",
  },
  Commit {
    id: a140cef0405e0bcbfb5de44ff59e091527d91b38,
    summary: "feat(parser): add ability to parse arrays",
  },
]
```



- amend
- as\_object
- author
- author\_with\_mailmap
- body
- body\_bytes
- committer
- committer\_with\_mailmap
- header\_field\_bytes
- id
- into\_object
- message
- message\_bytes
- message\_encoding
- message\_raw
- message\_raw\_bytes
- parent
- parent\_count
- parent\_id
- parent\_ids
- parents
- raw
- raw\_header
- raw\_header\_bytes
- summary
- summary\_bytes
- time
- tree
- tree\_id

```

1 /// Common commit object that is parsed from a repository.
2 #[derive(Debug, Default, Clone, PartialEq, Deserialize)]
3 #[serde(rename_all = "camelCase")]
4 pub struct Commit<'a> {
5     /// Commit ID.
6     pub id:          String,
7     /// Commit message including title, description and summary.
8     pub message:    String,
9     /// Conventional commit.
10    #[serde(skip_deserializing)]
11    pub conv:        Option<ConventionalCommit<'a>>,
12    /// Commit group based on a commit parser or its conventional type.
13    pub group:       Option<String>,
14    /// Default commit scope based on (inherited from) conventional type or a
15    /// commit parser.
16    pub default_scope: Option<String>,
17    /// Commit scope for overriding the default one.
18    pub scope:       Option<String>,
19    /// A list of links found in the commit
20    pub links:       Vec<Link>,
21    /// Commit author.
22    pub author:      Signature,
23    /// Committer.
24    pub committer:   Signature,
25 }

```

Step 3

```
1 // Generate changelog
2 let generate = |commits: Vec<git2::Commit>| -> Vec<String> {
3     commits
4         .iter()
5         .map(|commit| {
6             let id = commit.id().to_string();
7             format!(
8                 "* {} ([{}]({}))",
9                 commit.message().unwrap().trim(),
10                &id[0..7],
11                &id[0..7],
12            )
13        })
14        .collect()
15 };
16
17 let mut result = Vec::new();
18 result.push(String::from("## 0.1.0"));
19 result.push(String::from("\n### Bug Fixes\n"));
20 result.extend(generate(fixes));
21
22 result.push(String::from("\n### Features\n"));
23 result.extend(generate(features));
24 result.push(String::from("\n"));
```

```
1 // Print to file or stdout
2 use std::io::Write;
3 write!(std::io::stdout(), "{}", result.join("\n"))?;
```

## 0.1.0

### Bug Fixes

\* fix(args): rename help argument due to conflict ([9add0d4](9add0d4))

### Features

\* feat(cache): use cache while fetching pages ([df6aef4](df6aef4))

\* feat(config): support multiple file formats ([a9d4050](a9d4050))

\* feat(parser): add ability to parse arrays ([a140cef](a140cef))



<https://gist.github.com/orhun/e9992ebb1d7658b0cec1cbeab33a2bc2>



*smol!*  
git-cliff

### 1. Configuration

parse arguments

parse config file

### 2. Git Handling

process commits

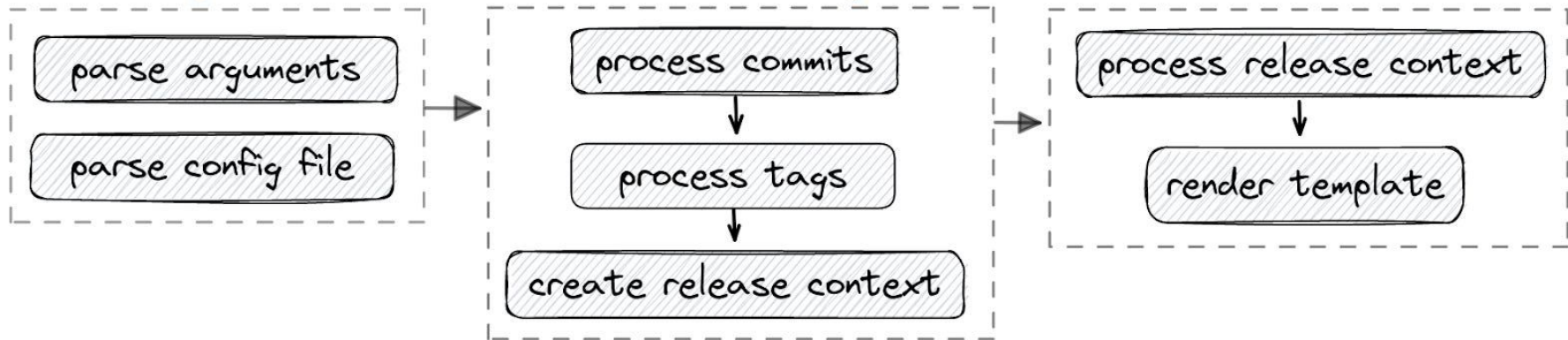
process tags

create release context

### 3. Generator

process release context

render template





- Argument parsing -> **clap**
- Configuration -> **TOML**
- Git handling -> **git2**
- Templating -> **Tera** (*Jinja2/Django*)

- \* df6aef4 (HEAD -> master) feat(cache): use cache while fetching pages
- \* a9d4050 feat(config): support multiple file formats
- \* 06412ac (tag: v1.0.1) chore(release): add release script
- \* e4fd3cf refactor(parser): expose string functions
- \* ad27b43 (tag: v1.0.0) docs(example)!: add tested usage example
- \* 9add0d4 fix(args): rename help argument due to conflict
- \* a140cef feat(parser): add ability to parse arrays
- \* 81fbc63 docs(project): add README.md
- \* a78bc36 Initial commit

```
{
  "version": "v1.0.0",
  "commits": [
    {
      "id": "81fbc6365484abf0b4f4b05d384175763ad8db44",
      "message": "add README.md",
      "body": null,
      "footers": [],
      "group": "Documentation",
      "breaking_description": null,
      "breaking": false,
      "scope": "project",
      "links": [],
      "author": {
        "name": "orhun",
        "email": "orhun@archlinux.org",
        "timestamp": 1626609601
      },
      "committer": {
        "name": "orhun",
        "email": "orhun@archlinux.org",
        "timestamp": 162661025
      },
      "conventional": true
    },
    {
      "id": "a140cef0405e0bcbfb5de44ff59e091527d91b38",
      "message": "add ability to parse arrays",
      ...
    }
  ]
}
```

# Tera

## Tera Basics

A Tera template is just a text file where variables and expressions get replaced with values when it is rendered. The syntax is based on Jinja2 and Django templates.

There are 3 kinds of delimiters and those cannot be changed:

- `{{` and `}}` for expressions
- `{%` and `%}` for statements
- `{#` and `#}` for comments

<https://keats.github.io/tera/>

```
{% if version %}\n    ## [{{ version | trim_start_matches(pat="v") }}] - [{{ timestamp | date(format="%Y-%m-%d") }}]\n{% else %}\n    ## [unreleased]\n{% endif %}\n{% if previous %}\n    {% if previous.commit_id %}\n        [{{ previous.commit_id | truncate(length=7, end="") }}]({{ previous.commit_id }})...\  
        [{{ commit_id | truncate(length=7, end="") }}]({{ commit_id }})\n    {% endif %}\n{% endif %}\n{% for group, commits in commits | group_by(attribute="group") %}\n    ### [{{ group | upper_first }}]\n    {% for commit in commits %}\n        - [{{ commit.message | upper_first }}] ([{{ commit.id | truncate(length=7, end="") }}]({{ commit.id }}))\  
        {% for footer in commit.footers -%}\n            , [{{ footer.token }}]{{ footer.separator }}[{{ footer.value }}]\n        {% endfor %}\n    {% endfor %}\n{% endfor %}\n
```

```
{% if version %}\n    ## [{{ version | trim_start_matches(pat="v") }}] - [{{ timestamp | date(format="%Y-%m-%d") }}]\n{% else %}\n    ## [unreleased]\n{% endif %}\n{% if previous %}\n    {% if previous.commit_id %}\n        [{{ previous.commit_id | truncate(length=7, end="") }}]({{ previous.commit_id }})...\  
        [{{ commit_id | truncate(length=7, end="") }}]({{ commit_id }})\n    {% endif %}\n{% endif %}\n{% for group, commits in commits | group_by(attribute="group") %}\n    ### [{{ group | upper_first }}]\n    {% for commit in commits %}\n        - [{{ commit.message | upper_first }}] ([{{ commit.id | truncate(length=7, end="") }}]({{ commit.id }}))\  
        {% for footer in commit.footers -%}\n            , [{{ footer.token }}]{{ footer.separator }}[{{ footer.value }}]\n        {% endfor %}\n    {% endfor %}\n{% endfor %}\n
```

```

{% if version %}\
    ## [{{ version | trim_start_matches(pat="v") }}] - [{{ timestamp | date(format="%Y-%m-%d") }}]
{% else %}\
    ## [unreleased]
{% endif %}\
{% if previous %}\
    {% if previous.commit_id %}
        [{{ previous.commit_id | truncate(length=7, end="") }}]({{ previous.commit_id }})... \
        [{{ commit_id | truncate(length=7, end="") }}]({{ commit_id }})
    {% endif %}\
{% endif %}\
{% for group, commits in commits | group_by(attribute="group") %}
    ### [{{ group | upper_first }}]
    {% for commit in commits %}
        - [{{ commit.message | upper_first }}] ([{{ commit.id | truncate(length=7, end="") }}]({{ commit.id }})) \
        {% for footer in commit.footers -%}
            , [{{ footer.token }}]{{ footer.separator }}[{{ footer.value }}] \
        {% endfor %}\
    {% endfor %}
{% endfor %}\n

```

```

{% if version %}\
    ## [{{ version | trim_start_matches(pat="v") }}] - [{{ timestamp | date(format="%Y-%m-%d") }}]
{% else %}\
    ## [unreleased]
{% endif %}\
{% if previous %}\
    {% if previous.commit_id %}
        [{{ previous.commit_id | truncate(length=7, end="") }}]({{ previous.commit_id }})... \
        [{{ commit_id | truncate(length=7, end="") }}]({{ commit_id }})
    {% endif %}\
{% endif %}\
{% for group, commits in commits | group_by(attribute="group") %}
    ### [{{ group | upper_first }}]
    {% for commit in commits %}
        - [{{ commit.message | upper_first }}] ([{{ commit.id | truncate(length=7, end="") }}]({{ commit.id }})) \
        {% for footer in commit.footers -%}
            , [{{ footer.token }}]{{ footer.separator }}[{{ footer.value }}] \
        {% endfor %}\
    {% endfor %}
{% endfor %}\n

```



```
{% macro commit(commit) -%}
  - {% if commit.scope %}*({{ commit.scope }})* {% endif %}{% if commit.breaking %}! {% endif %}\
    {{ commit.message }} - ({{ commit.id }}(<REPO>/commit/{{ commit.id }}))\
{% endmacro -%}
```

```
# regex for parsing and grouping commits
commit_parsers = [
  { message = "^feat", group = "Features" },
  { message = "^fix", group = "Bug Fixes" },
  { message = "^doc", group = "Documentation" },
  { message = "^perf", group = "Performance" },
  { message = "^refactor", group = "Refactor" },
  { message = "^style", group = "Styling" },
  { message = "^test", group = "Testing" },
  { message = "^chore\\(deps\\)", skip = true },
  { message = "^chore\\(pr\\)", skip = true },
  { message = "^chore\\(pull\\)", skip = true },
  { message = "^chore\\(release\\): prepare for", skip = true },
  { message = "^chore|ci", group = "Miscellaneous Tasks" },
  { body = ".*security", group = "Security" },
]
```

## [1.0.1] - 2021-07-18

[ad27b43](ad27b43e8032671afb4809a1a3ecf12f45c60e0e)...[06412ac]  
(06412ac1dd4071006c465dde6597a21d4367a158)

### Miscellaneous Tasks

- Add release script ([06412ac](06412ac1dd4071006c465dde6597a21d4367a158))

### Refactor

- Expose string functions ([e4fd3cf](e4fd3cf8e2e6f49c0b57f66416e886c37cbb3715))

## [1.0.0] - 2021-07-18

### Bug Fixes

- Rename help argument due to conflict ([9add0d4](9add0d4616dc95a6ea8b01d5e4d233876b6e5e00))

### Documentation

- Add README.md ([81fbc63](81fbc6365484abf0b4f4b05d384175763ad8db44))
- Add tested usage example ([ad27b43](ad27b43e8032671afb4809a1a3ecf12f45c60e0e))

### Features

- Add ability to parse arrays ([a140cef](a140cef0405e0bcbfb5de44ff59e091527d91b38))

# Changelog [↗](#)

---

All notable changes to this project will be documented in this file.

## 1.4.0 - 2023-10-29

---

### Features [↗](#)

- (*changelog*) Support bumping the semantic version via `--bump` ([#309](#)) - ([bcfcd1f](#))
- (*ci*) Add 'typos' check ([#317](#)) - ([88c34ab](#))
- (*command*) Log the output of failed external commands - ([205cdbb](#))
- (*config*) [**breaking**] Support regex in 'tag\_pattern' configuration ([#318](#)) - ([3c2fb60](#))
- (*config*) Add field and value matchers to the commit parser ([#312](#)) - ([04fbc8](#))

### Documentation [↗](#)

- (*blog*) Fix the TOML format in 1.4.0 blog post - ([4d691d2](#))
- (*blog*) Add blog post for 1.4.0 release - ([e3f1b3b](#))
- (*changelog*) Fix typos ([#316](#)) - ([edd3c30](#))
- (*config*) Update the comment for tag\_pattern - ([596fd4d](#))

## 0.24.0 - 2023-10-23

We are excited to announce the new version of `ratatui` - a Rust library that's all about cooking up TUIs 🍳

In this version, we've introduced features like window size API, enhanced chart rendering, and more. The list of *breaking changes* can be found [here](#) ⚠️. Also, we created various tutorials and walkthroughs in [Ratatui Book](#) which is available at <https://ratatui.rs> 🚀

🌟 **Release highlights:** <https://ratatui.rs/highlights/v0.24.html>

### Features [↗](#)

- [c6c3f88](#) (*backend*) Implement common traits for `WindowSize` (#586)
- [d077903](#) (*backend*) Backend provides `window_size`, add `Size` struct (#276)

For `image` (`sixel`, `iTerm2`, `Kitty`...) support that handles graphics in terms of `Rect` so that the image area can be included in layouts.

For example: an image is loaded with a known pixel size, and drawn, but the image protocol has no mechanism of knowing the actual cell/character area that been drawn on. It is then impossible to skip overdrawing the area.

Returning the window size in `pixel-width / pixel-height`, together with `columns / rows`, it can be possible to account the pixel size of each cell / character, and then known the `Rect` of a given image, and also resize the image so that it fits exactly in a `Rect`.



Integration 

```
cargo install git-cliff
```

```
git-cliff --init
```

```
vim cliff.toml
```

```
git-cliff -o CHANGELOG.md
```

# cliff.toml

```
[changelog]
# changelog header
header = ""
# template for the changelog body
# https://keats.github.io/tera/docs/#introduction
body = ""
# remove the leading and trailing whitespace from the template
trim = true
# changelog footer
footer = ""
# postprocessors
postprocessors = [
  { pattern = '<REPO>', replace = "https://github.com/orhun/git-cliff" }, # replace repository URL
]
```



```
1 [git]
2 # parse the commits based on https://www.conventionalcommits.org
3 conventional_commits = true
4 # filter out the commits that are not conventional
5 filter_unconventional = true
6 # process each line of a commit as an individual commit
7 split_commits = false
8 # regex for preprocessing the commit messages
9 commit_preprocessors = [
10  { pattern = '\\((\\w+\\s)?#[0-9+\\s)\\)', replace = "([#${2}](<REPO>/issues/${2}))" },
11  # Check spelling of the commit with https://github.com/crate-ci/typos
12  # If the spelling is incorrect, it will be automatically fixed.
13  { pattern = '.*', replace_command = 'typos --write-changes -' },
14 ]
15 # regex for parsing and grouping commits
16 commit_parsers = [
17  { message = "^feat", group = "<!-- 0 --> 🏔 Features" },
18  { message = "^fix", group = "<!-- 1 --> 🐛 Bug Fixes" },
19 ]
```

```
1 [git]
2 # protect breaking changes from being skipped due to matching a skipping
  commit_parser
3 protect_breaking_commits = false
4 # filter out the commits that are not matched by commit parsers
5 filter_commits = false
6 # regex for matching git tags
7 tag_pattern = "v[0-9].*"
8 # regex for skipping tags
9 skip_tags = "beta|alpha"
10 # regex for ignoring tags
11 ignore_tags = "rc"
12 # sort the tags topologically
13 topo_order = false
14 # sort the commits inside sections by oldest/newest order
15 sort_commits = "newest"
```

*# generate changelog for unreleased commits*

```
git cliff --unreleased
```

*# set a tag for the unreleased changes*

```
git cliff --tag 1.0.0
```

*# calculate and set the next semantic version*

```
git cliff --bump
```

*# generate a changelog scoped to a specific directory*

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

*# generate changelog for a specific commit range*

```
git cliff 4c7b043..a440c6e
```

*# print context to stdout as JSON*

```
git cliff --context
```

```
# generate changelog for unreleased commits
```

```
git cliff --unreleased
```

```
# set a tag for the unreleased changes
```

```
git cliff --tag 1.0.0
```

```
# calculate and set the next semantic version
```

```
git cliff --bump
```

```
# generate a changelog scoped to a specific directory
```

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

```
# generate changelog for a specific commit range
```

```
git cliff 4c7b043..a440c6e
```

```
# print context to stdout as JSON
```

```
git cliff --context
```

*# generate changelog for unreleased commits*

```
git cliff --unreleased
```

*# set a tag for the unreleased changes*

```
git cliff --tag 1.0.0
```

*# calculate and set the next semantic version*

```
git cliff --bump
```

*# generate a changelog scoped to a specific directory*

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

*# generate changelog for a specific commit range*

```
git cliff 4c7b043..a440c6e
```

*# print context to stdout as JSON*

```
git cliff --context
```

*# generate changelog for unreleased commits*

```
git cliff --unreleased
```

*# set a tag for the unreleased changes*

```
git cliff --tag 1.0.0
```

*# calculate and set the next semantic version*

```
git cliff --bump
```

*# generate a changelog scoped to a specific directory*

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

*# generate changelog for a specific commit range*

```
git cliff 4c7b043..a440c6e
```

*# print context to stdout as JSON*

```
git cliff --context
```

*# generate changelog for unreleased commits*

```
git cliff --unreleased
```

*# set a tag for the unreleased changes*

```
git cliff --tag 1.0.0
```

*# calculate and set the next semantic version*

```
git cliff --bump
```

*# generate a changelog scoped to a specific directory*

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

*# generate changelog for a specific commit range*

```
git cliff 4c7b043..a440c6e
```

*# print context to stdout as JSON*

```
git cliff --context
```

*# generate changelog for unreleased commits*

```
git cliff --unreleased
```

*# set a tag for the unreleased changes*

```
git cliff --tag 1.0.0
```

*# calculate and set the next semantic version*

```
git cliff --bump
```

*# generate a changelog scoped to a specific directory*

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

*# generate changelog for a specific commit range*

```
git cliff 4c7b043..a440c6e
```

*# print context to stdout as JSON*

```
git cliff --context
```



*# generate changelog for unreleased commits*

```
git cliff --unreleased
```

*# set a tag for the unreleased changes*

```
git cliff --tag 1.0.0
```

*# calculate and set the next semantic version*

```
git cliff --bump
```

*# generate a changelog scoped to a specific directory*

```
git cliff --include-path "**/*.toml" --exclude-path ".github/*"
```

*# generate changelog for a specific commit range*

```
git cliff 4c7b043..a440c6e
```

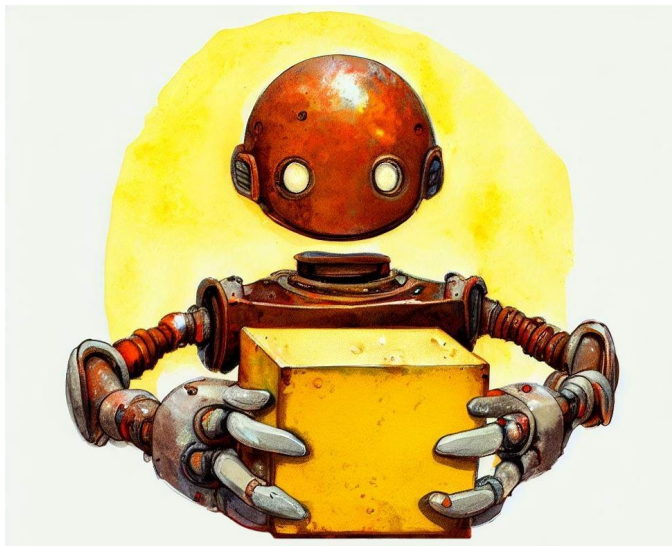
*# print context to stdout as JSON*

```
git cliff --context
```

As a library 🦀

# release-plz

Release Rust crates from CI with a Release PR 🤖 by @Marcoleni



<https://github.com/Marcoleni/release-plz>

```
[workspace.dependencies]
```

```
git-cliff-core = { version = "1.3.1", default-features = false }
```

```
use git_cliff_core::{  
    changelog::Changelog as GitCliffChangelog,  
    commit::Commit,  
    config::{ChangelogConfig, CommitParser, Config, GitConfig},  
    release::Release,  
};
```

```
1 impl Changelog<'_> {
2     /// Generate the full changelog.
3     pub fn generate(self) -> String {
4         let config = self
5             .config
6             .unwrap_or_else(|| default_git_cliff_config(None, self.release_link.as_deref()));
7
8         let changelog = GitCliffChangelog::new(vec![self.release], &config)
9             .expect("error while building changelog");
10
11         let mut out = Vec::new();
12         changelog
13             .generate(&mut out)
14             .expect("cannot generate changelog");
15         String::from_utf8(out).expect("cannot convert bytes to string")
16     }
17 }
```

# Automated Rust Releases



<https://blog.orhun.dev/automated-rust-releases/>

Future 

## 38 Oxidization

### ☰ git-cliff

- walk commit graph topo + time sorted
- extract commit information
- walk tags and resolve them
- filter by changes to a particular path

<https://github.com/orhun/git-cliff>

Added by Byron



[HOME](#)[ABOUT](#)[GITHUB](#)

git-cliff can generate [changelog](#) files from the [Git](#) history by utilizing [conventional commits](#) as well as regex-powered custom parsers. The [changelog template](#) can be customized with a [configuration file](#) to match the desired format.



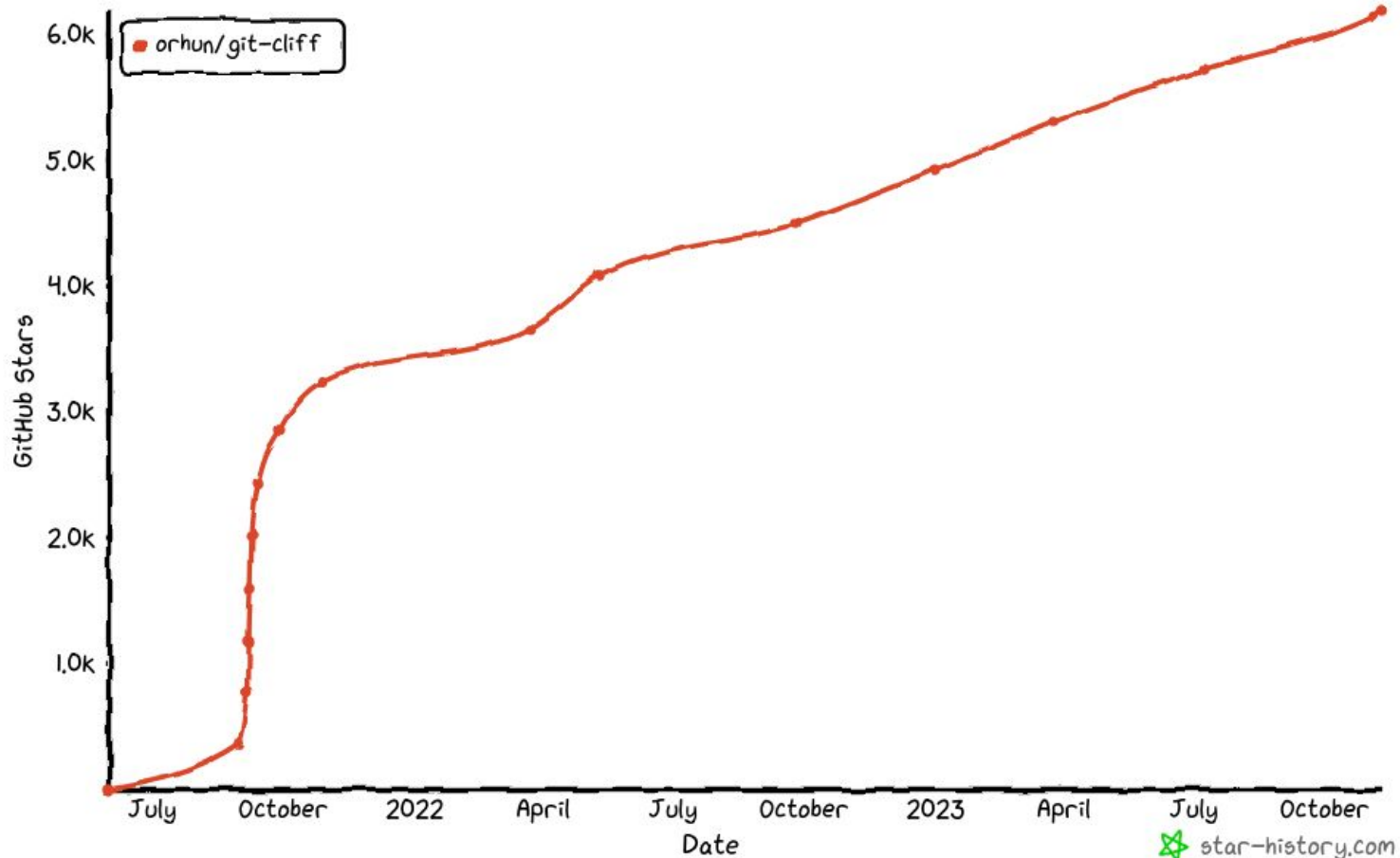
On this web version, you can provide a Git repository URL and generate changelogs by tweaking the settings. Feel free to try below!

#### Config

```
# git-cliff ~ default configuration file
# https://git-cliff.org/docs/configuration
#
```

#### Result

# Star History



## Become a sponsor to **Orhun Parmaksız**



### Orhun Parmaksız

orhun

Ankara, Turkey

Thank you for considering supporting my work! ✨

#### 🦀 I am involved in:

- My own projects including [git-cliff](#), [gpg-tui](#), [daktilo](#) and more!
- [Arch Linux](#): a lightweight and flexible Linux distribution that tries to *Keep It Simple*.
  - I am a package maintainer for 2 years and I maintain 150+ packages in the official repositories.
  - My primary goal is to improve the Rust ecosystem for users. I work closely with upstream sources to patch projects and add new features.
- [Alpine Linux](#): a security-oriented, lightweight Linux distribution.
  - I'm actively maintaining Rust packages.
- [Spicy Lobster Studio](#): a gamedev company built with openness, kindness, and curiosity.
  - I work on infrastructure, build-related issues, and releases of the open source games.
- [Ratatui](#): the official successor of [tui-rs](#), a Rust library that's all about cooking up terminal user interfaces.
  - I'm taking an active part as a project maintainer.

85% towards 20 monthly sponsors goal



developer-guy and 16 others sponsor this goal

Sponsor as orhun ▾



Hover over your avatar to review the badge you'll get that shows [@orhun](#) you're a sponsor.

Select a tier

Monthly

One-time

\$ 5



a month

Select

You'll receive any rewards listed in the \$5 monthly tier.

**\$1 a month**

Select

🦀 Every bit helps!

**\$5 a month**

Select

<https://github.com/sponsors/orhun>



lloydjatkinson



jjerphan



Psyhackological



Xenira



wcampbell0x2a



vcrn



log101



kdy1



erlend-sh



aliereno



Lissy93



Itabis



mstg



Marcoleni



barbarosaffan



hituzi-no-sippo



developer-guy



kakkoyun



Anton Hvornum



le\_chuck



Sven Assmann

# Thank you!

orhun.dev  
git-cliff.org