

Versioning your writing

Shane Crowley

Writing is iterative. It normally involves errors, tangents and refinements. A submitted piece of writing is the culmination of this process. Versioning allows you to take snapshots of your writing as it evolves. This keeps a record of your thinking, tracks contributions from team members, and makes it possible to switch to an earlier version at any time.

June 16, 2025

Shane is a former University College Cork academic.
He now works in software as a technical author.
You can contact him at edibotopic@gmail.com.

Contents

1. Using Google Docs to version your project	4
1.1. Getting started	4
1.2. Changes are autosaved	4
1.3. A document has a history	4
1.4. Restoring an old version	5
1.5. Versions are useful	5
1.6. Naming your versions	6
1.7. Sharing a file with its version history	7
1.8. Collaboration history	8
2. Alternatives to Google Docs	10
2.1. Microsoft Word and Office365	10
2.2. Git and GitHub (advanced)	10
2.2.1. Prerequisites	10
2.2.2. Git workflow example	11
2.2.3. GitHub workflow example	12
2.2.4. Aside: Why Git?	13
2.2.5. Appendix: Example of a Pull Request	14

1. Using Google Docs to version your project

There are many ways to version a project. I will explain the concept using Google Docs. The workflow for Microsoft Word is similar and is briefly described in Section 2.1.

1.1. Getting started

You will need a [Google account](#) to follow this guide¹.

To start writing, go to docs.google.com and **Start a new document**.

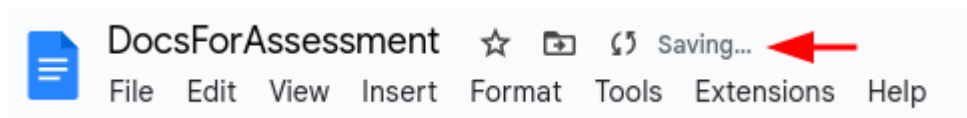
You can also go to drive.google.com, click **New** and select **Google Docs**.

Google Docs is free, online-first software. Files are edited in the browser and stored in the cloud, with automatic backups. It makes it easy to access your files across devices and collaborate on your project with other people.

1.2. Changes are autosaved

Start making some changes to the file you created.

Notice that your file is saved after each change:



Each time a file is saved, it is also stored in the **version history** of the file.

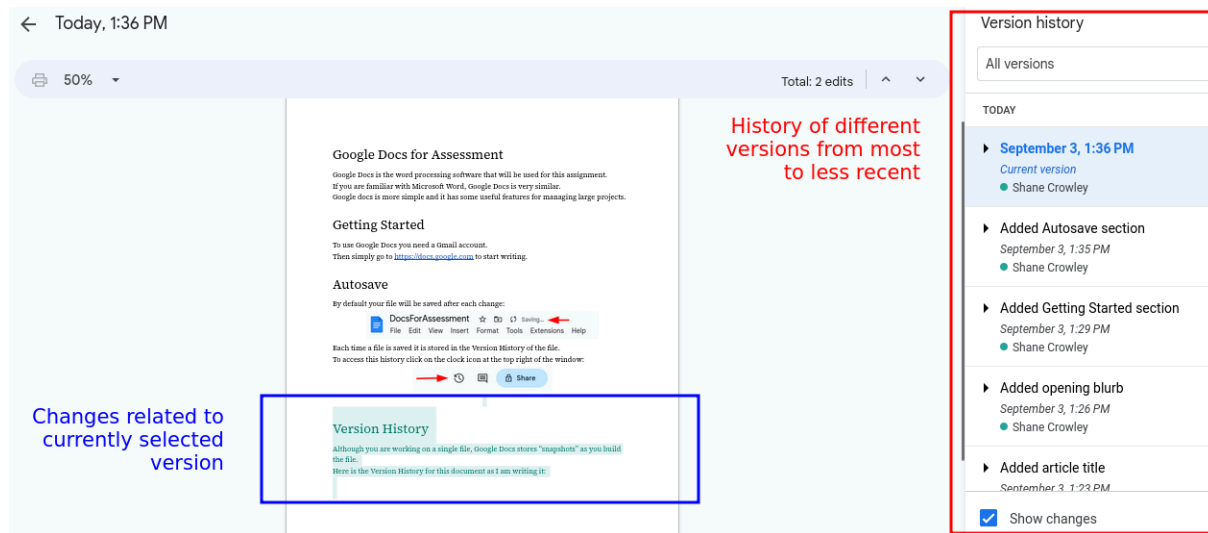
1.3. A document has a history

To access the history for your file, left click on the clock icon at the top right of your screen:



Here is some of the history for what you are reading, as I wrote it:

¹Existing (non-gmail) accounts can also be linked with Google, but I have not tested this.



The history consists of a sequence of **versions**.

Select a version to see that version and the changes it introduced.

The most recent change in this example was adding a section on “Version History”, which is highlighted in green.

Make some different changes to your file and check the versions.

Additions are highlighted in **green** and deletions in **red**.

1.4. Restoring an old version

At any point you can *restore* a previous version.

This is one of the reasons versioning is popular in the software industry. If a bug is introduced in the latest version, it is always possible to go back in time before the bug was introduced.

To restore an old version, find the version you want in the history, **left click** on the **⋮** icon, then select **Restore this version**.

1.5. Versions are useful

Versions are useful, whether you are writing as an individual or in a team.

1. **Restoring previous versions after an error:** it’s common to make an edit that you later regret, such as deleting an image that you later need but can’t find. A version history gives you the security of a time machine.
2. **Checking the workload between collaborators:** when editing a file with collaborators (team projects), the contribution of different people are highlighted in each version. If there is ever a question about the balance of workload, the version history can be consulted.
3. **Providing evidence of thoughtful and consistent work:** if you are a student, you want your submission to be a faithful record of your hard work. *You didn’t just thoughtlessly*

copy-and-paste an AI-generated response a day before the deadline, did you? When sharing a Google Doc, you can also include its version history, which shows how your project evolved.

The typical alternative to versioning is to create separate files like `myProjectDraft1.doc`, `myProjectDraft2.doc`, `myProjectDraft3.doc`. This gets complicated fast and doesn't provide an easy way to compare across drafts.

1.6. Naming your versions

When Google Docs autosaves, the default name of the version is the current date and time.

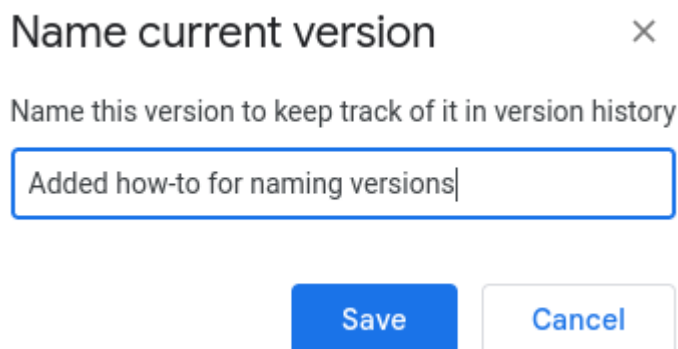
It is also possible to give your version a unique name².

Done right, named versions clearly express the nature of the change that has been made.

A history with named versions can be a clear record of the types of decisions that you made and why you made them³.

This can help you to maintain discipline as you write. Later, if you forget why exactly you did something, you can consult the history.

When you are ready to name a version select `File > Version history > Name current version`. You can then enter a short description:



Name current version ×

Name this version to keep track of it in version history

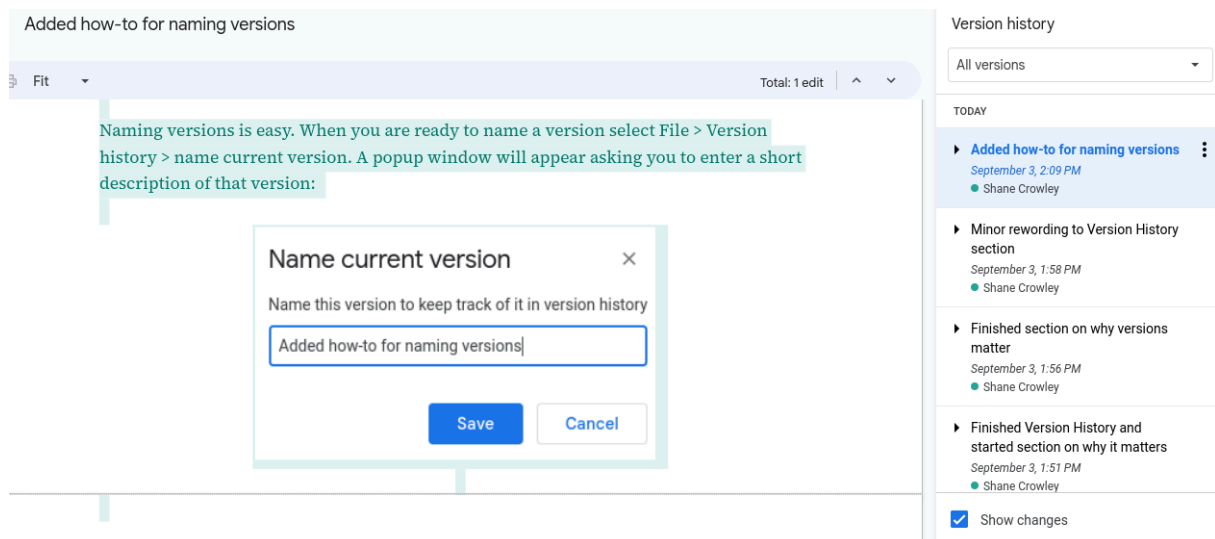
Added how-to for naming versions|

Save Cancel

This will then appear as an entry in the version history.

²There is a **limit of 40 named versions** per file, which should be enough for small projects.

³It's best to use named versions for substantial changes (e.g., rewrote section for historical accuracy) rather than trivial fixes (e.g., fixed a typo)

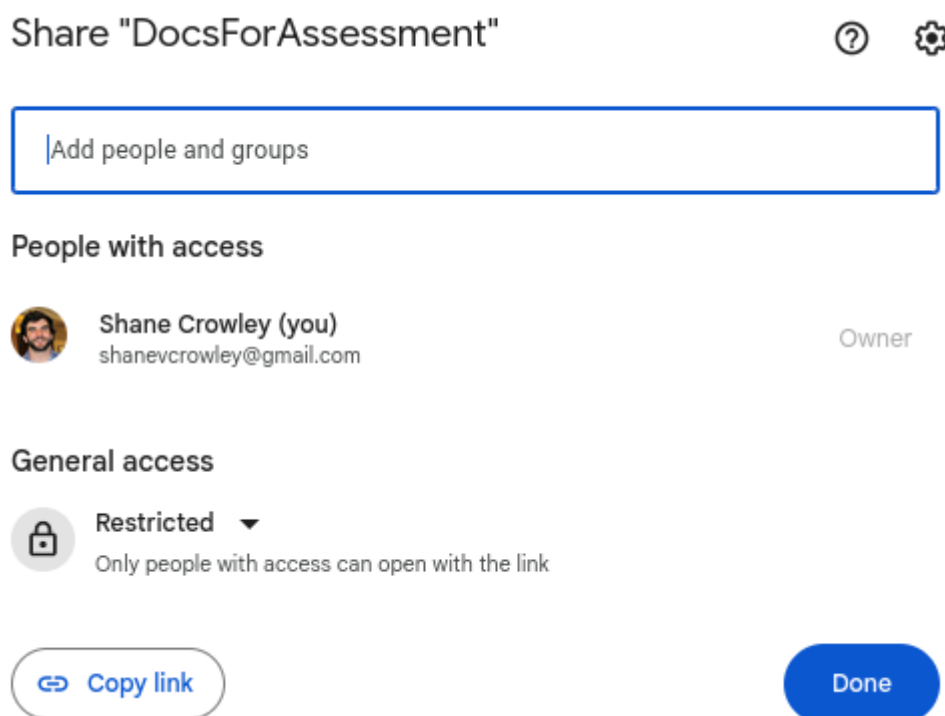


1.7. Sharing a file with its version history

You often need to share a file with collaborators and assessors.

Click the large **Share** button at the top right of the window.

The popup shows who already has access and gives you a prompt to add more people by typing their email address:



After you type an email, a dropdown menu allows you to assign user privileges.

← Share "DocsForAssessment" ? ⚙


edibotopic@gmail.com X

Editor ▼

☒ Notify people

Message

Please collaborate on this file.

 Cancel Send



Typically, you want to grant full access rights as an “Editor”:

If you click on **Share** again subsequently you will see that the person has been added:

Share "DocsForAssessment" ? ⚙

Add people and groups

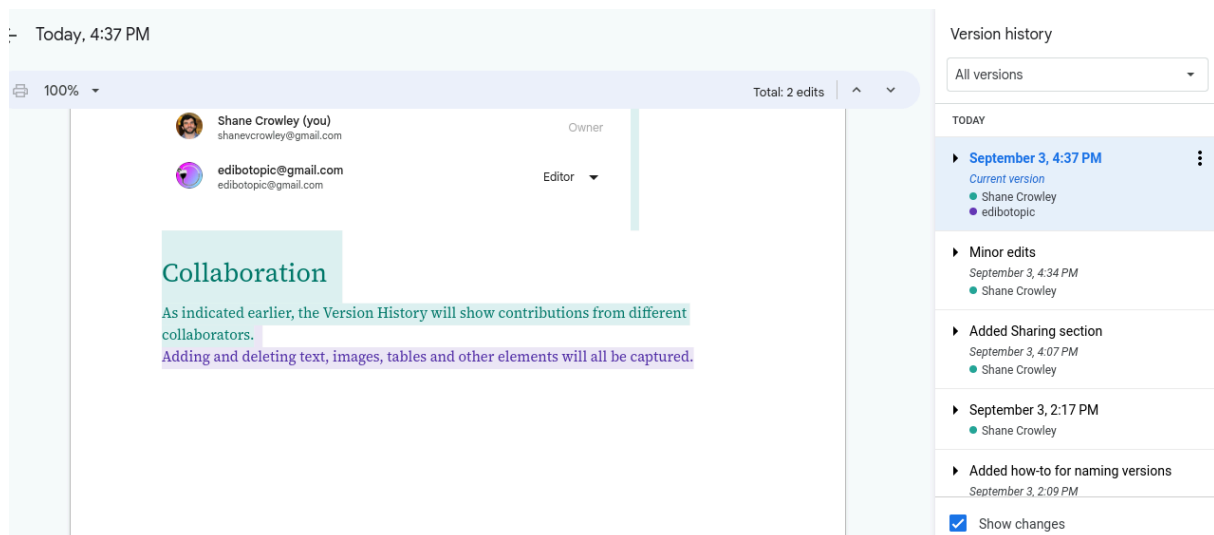
People with access

	Shane Crowley (you) shanevcrowley@gmail.com	Owner
	edibotopic@gmail.com edibotopic@gmail.com	Editor ▼

1.8. Collaboration history

The Version History shows contributions from different collaborators who have edit access to the file.

When someone adds or deletes text, images, tables and other elements, the change is visually associated with the individual contributor.



In the screenshot above, the most recent changes were made by two collaborators⁴.

Their names are listed in the version and the changes are **colour-coded**, allowing the changes they made to be identified.

In a collaborative project, the version history can help reveal disparities in workload. It can also help identify the areas in which people made significant contributions.

⁴Use an identifiable/professional display name. If someone was checking the work above, they may not be able to identify who “edibotopic” is. The display name can be changed in your Google account settings

2. Alternatives to Google Docs

There are many alternatives to Google Docs.

Some, like Microsoft Word, offer a virtually identical workflow.

Others, like Git, are powerful but more suitable for advanced users.

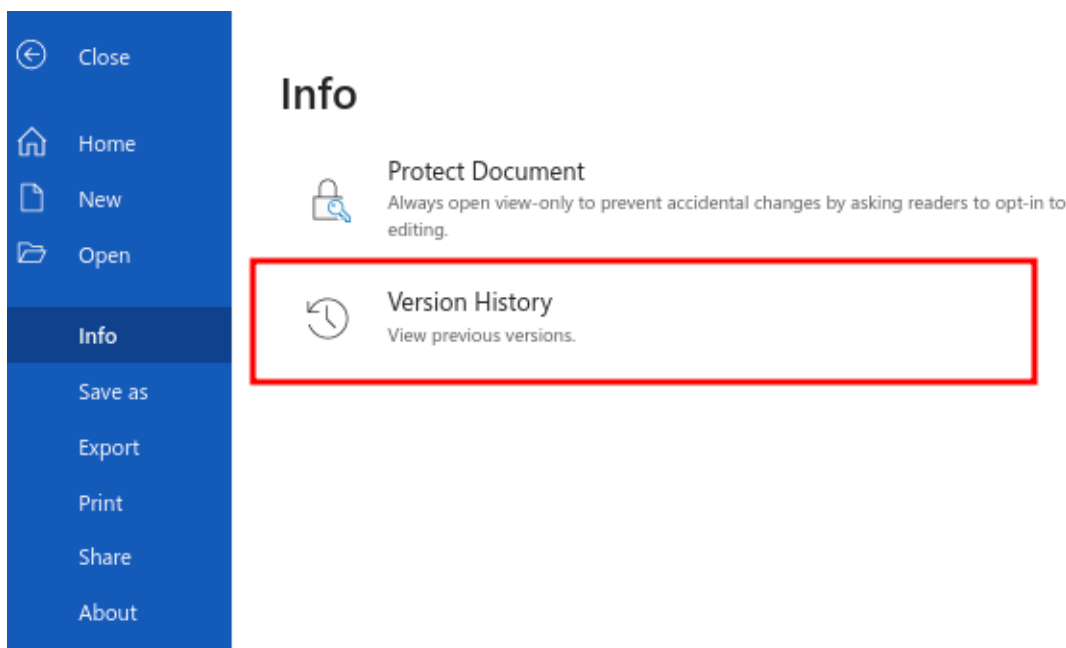
In an assessment scenario, Google Docs or Microsoft Word should work for most students. Students are ultimately responsible for ensuring that their version history is accessible, regardless of the tool they use.

2.1. Microsoft Word and Office365

The online version of Word through Office365 also has a version history.

You can access it through `File > Info > Version History`.

This works similarly to Google Docs but one limitation is that you **cannot name versions**.



2.2. Git and GitHub (advanced)

Git is the most popular version control tool in the world. It was invented by Linus Torvalds, who also created Linux, and has been widely used for over 20 years.

GitHub is an online service for hosting projects versioned with Git. People use GitHub to publish the source of software applications but also books, courses, and laboratory websites.

Git can be used on its own or with GitHub and other similar online services, like GitLab.

2.2.1. Prerequisites

- Familiarity with using the terminal or a willingness to learn.

- A Mac or Linux device. Windows can also be used with WSL or Git bash.

2.2.2. Git workflow example

This example assumes you are using Debian or Ubuntu on Linux/WSL.

Note: some steps, like creating a username and email, are excluded. Git's error messages provide clear instructions for how to take these steps when they are needed.

In a terminal, first ensure that Git is installed:

```
sudo apt install git
```

Make a directory for your project and change into that directory:

```
mkdir myProject
cd myProject
```

Initialise the project as a Git repository:

```
git init
```

Create a markdown file to start editing:

```
touch article.md
```

The files in the directory are currently not tracked. To change that, add all the files:

```
git add .
```

Next, commit the changes and add a message describing the commit:

```
git commit -m "Start of the project."
```

Edit the file using your text editor of choice, such as Vim, Nano, VS Code or Sublime Text. I will use Vim because it runs in the terminal:

```
vim article.md
```

Add something, like a header:

```
# My interesting project
```

Save the file.

If you want to check your progress, run:

```
git status
```

This should show you that there are changes in your `article.md` file.

If you want to see what the changes are you can check the **difference**:

```
git diff article.md
```

This should show you lines that were added (or deleted):

```
+ # My interesting project
```

Commit the changes, again adding an appropriate description:

```
git commit -m "Added a header."
```

You have now made two commits, which are tracked by Git in your version history.

To view the history:

```
git log
```

2.2.3. GitHub workflow example

Create a GitHub account if you don't already have one.

To create a new remote repository, click the green **New** button or the plus sign on GitHub.

After giving your repository a name, find the snippet for pushing an existing repository, which should look like this:

```
git remote add origin git@github.com:my-profile-name/my-project-name.git
git branch -M main
git push -u origin main
```

All of these commands should run OK, except the last one. See the note below for resolving this issue.

For security reasons, GitHub requires SSH keys before you can push files to your account.

This takes some setup but **you only need to do it once**.

Create an ssh key:

```
ssh-keygen -t rsa -b 4096 -C "my_email@example.com"
```

This creates a public key in `~/.ssh/id_rsa.pub`.

Open that file and copy the contents.

Make sure you are logged in to GitHub, and go to <https://github.com/settings/keys>.

Select **New SSH key**, enter a title and paste the key.

With SSH keys set up, you can now run:

```
git push -u origin main
```

Go to the GitHub repository, refresh the page, and you should see `article.md`.

Now, each time you commit some changes, you can `git push` to update the remote repository on GitHub.

If you have Git set up on another device, you can `git clone` to copy the GitHub repository and then `git push` when you have new changes.

2.2.4. Aside: Why Git?

The above is a minimal description of what Git and GitHub can do. They are far more powerful than I am able to communicate in this document.

It might seem like a lot of extra work but there is a reason millions of people working in programming, data science and software documentation use it.

Generally, it allows for **more granular** control of your versioning. For example, if you make multiple changes in one file, or across multiple files, Git allows you to group specific changes into different commits with their own descriptions.

Git also has a powerful feature called **branching**. If someone on your team has a big new idea, they can create a branch from the project where they can work on that idea in isolation, without interfering with the main project.

If a Git branch is pushed to GitHub, it generates a **Pull Request** — essentially this means: “I have made some changes, and I am requesting that you consider pulling those changes into the project”. At that point, the team can formally review the changes. Helpfully, GitHub highlights where the changes happened and makes it easy to accept/reject specific changes.

Automations can also be set up on GitHub; for example, submissions can be automatically checked for spelling, broken links, and so on. If you are writing a book, you can set GitHub up to automatically render it in different formats (e.g., `.pdf`, `.epub`, `.html`) when you update its contents.

Git works on a variety of files for text and code. Google Docs only works with Google Docs files. For writers, **plaintext** formats like markdown work best, and they can later be converted into Google Docs or Microsoft Word formats if you need.

Although it initially seems difficult, Git is more **teachable** than something like Google Docs. Why? I can write in text the commands that are needed to do something, without needing screenshots or video — someone can just copy the text and try it out. Git has worked the same way for over 20 years and is unlikely to change (unlike Google Docs). There is a vast amount of documentation online about how to use Git. Git is also a valuable skill — many jobs in data science and software expect some basic proficiency with Git.

The basics of Git can be learned in a weekend but its advanced features can take much longer to understand. If you are afraid of the terminal, text editors like VSCode offer a visual, button-based approach, which can be good enough in many cases.

If you want to know more about using Git or GitHub for your project, contact me at edibotopic@gmail.com.

2.2.5. Appendix: Example of a Pull Request

Below is an example of a Pull Request review for one of the open-source projects that I work on.

I was developing some new documentation for a project and asked for help from the community.

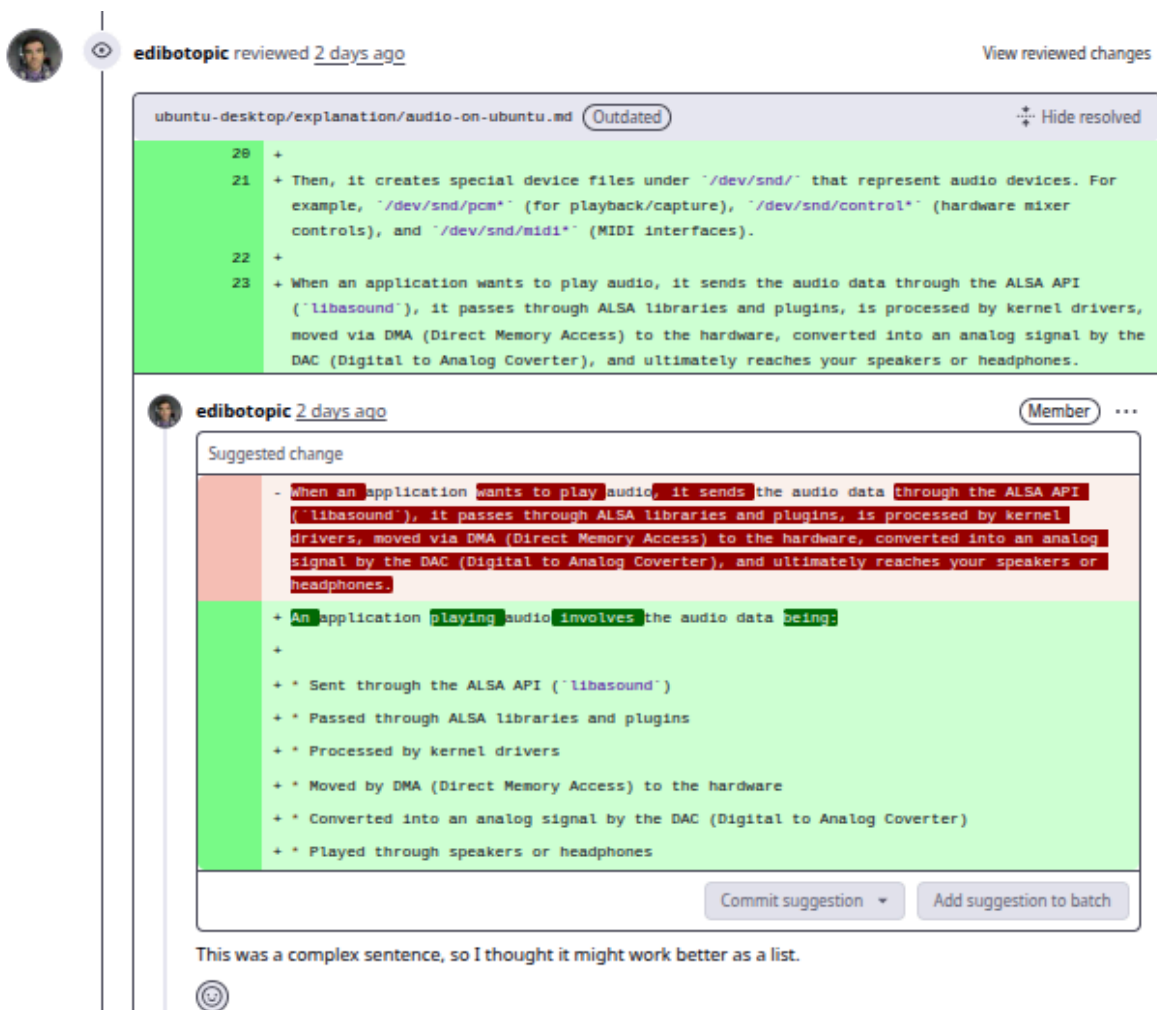
A contributor cloned the GitHub repository, created a branch to make their changes and then wrote the article.

Later, they pushed their branch to create a Pull Request.

I then read the article and provided suggestions, which the contributor could either accept or reject.

The entire history of this process, including versions, changes and comments, is available publicly.

The community member is credited as a contributor to the project and can point to their contribution.



edibotopic reviewed 2 days ago [View reviewed changes](#)

ubuntu-desktop/explanation/audio-on-ubuntu.md Outdated Hide resolved

20 +
21 + Then, it creates special device files under `/dev/snd/` that represent audio devices. For
example, `/dev/snd/pcm*` (for playback/capture), `/dev/snd/control*` (hardware mixer
controls), and `/dev/snd/midi*` (MIDI interfaces).
22 +
23 + When an application wants to play audio, it sends the audio data through the ALSA API
(`libasound`), it passes through ALSA libraries and plugins, is processed by kernel
drivers, moved via DMA (Direct Memory Access) to the hardware, converted into an analog
signal by the DAC (Digital to Analog Converter), and ultimately reaches your speakers or
headphones.

edibotopic 2 days ago Member ...

Suggested change

- When an application wants to play audio, it sends the audio data through the ALSA API
(`libasound`), it passes through ALSA libraries and plugins, is processed by kernel
drivers, moved via DMA (Direct Memory Access) to the hardware, converted into an analog
signal by the DAC (Digital to Analog Converter), and ultimately reaches your speakers or
headphones.

+ An application playing audio involves the audio data being:

- + Sent through the ALSA API (`libasound`)
- + Passed through ALSA libraries and plugins
- + Processed by kernel drivers
- + Moved by DMA (Direct Memory Access) to the hardware
- + Converted into an analog signal by the DAC (Digital to Analog Converter)
- + Played through speakers or headphones

[Commit suggestion](#) [Add suggestion to batch](#)

This was a complex sentence, so I thought it might work better as a list.