syncoxide.rs a cloud file sync, sharing, backup and encryption solution

The purpose of this project is to offer an easy and reliable way to sync files between multiple providers (like Google Drive, Dropbox, S3, SFTP servers, ...) and local files, and a simple and quick way for file sharing, backup and encryption.

It offers real time sync (from simple Copy One-way to Two-way Sync) all handled in the cloud, without the explicit need of local clients.

Use cases

- You have various cloud providers and you need to keep files in sync between them
- You want to quickly share a local file with someone with minimal tools involved
- You want to share a Dropbox file with someone that doesn't have a Dropbox account but want to keep that file in sync with them
- Someone want to send you a file but doesn't have any cloud providers accounts
- Keep local files in sync between multiple devices, without any other storage providers, directly with P2P
- You want a global view of files among all providers, My Drive, shared, starred, recents and trash folders
- You want to access your remote providers files locally in the filesystem
- Encrypt local files or files stored on remote providers
- Search capabilities
- Analytics, what files you most often access, what types of files do you have, ...
- Quick and secure backups
- Photos manager

Features

- Sync between providers, Copy, Sync, Move, Two-way sync
- Conflict resolution
- Sync local files between multiple authenticated devices
- MD5, SHA1 hashes are checked at all times for file integrity
- Multi-threaded transfers
- Make sync (and changes) with WAL (Write-Ahead Logging) to ensure file integrity
- Share between multiple providers like Google Drive, Dropbox, S3, ...
- Share local files

- Easy way to share files with external users, via a link or notification.
- Handles very large files efficiently with concurrent and resumable transfers
- Receiving files, like S3 presigned URLs but create a dest folder where others can upload more files and even folders
  - deadline, late uploads, password
- Encrypted files and folders, with files saved in provider or with local files
  - also encrypted sharing with PGP
- Browser app built in WebAssembly compatible with all major browsers
- Local app for desktop and mobile, access files via FUSE on Linux and macOS, WinFSP (or others) on Windows and file picker on mobile
  - encrypted cache, full copy kept in sync, notifications
- Local views: My Drive, Computers, Shared, Shared with me, Recents, Starred, Trash
- Global view from all providers and local files, My Drive, Computers, Shared, Shared with me, Recents, Starred, Trash
- Internal links to other files
- Search capabilities
- Analytics
- File history and versioning
- Integration with other systems
- Cleanup storage
- Sync, share status, storage overview
- Backups with borg, encrypted, deduplicated, compressed
- Automation, convert to PDF, convert image, unzip, convert audio/video, watermark files
- Photos manager
- REST API, gRPC, CLI clients and client libs in multiple languages
- Many supported providers

What separates it from other products

Compared to S3Drive, rclone, Resilio Sync, Nextcloud, Syncthing, ownCloud, Seafile

- Sync and Share between cloud providers
- Simple and quick way to share files with someone using minimal tools, ideally only by browser
- Receiving files, like S3 presigned URLs but create a dest folder where others can upload more files and even folders
- Can combine both local files Sync and Share with remote providers
- rclone is not quite real time Sync from remote to local, it has delay for auto-sync (even if you are listing the content of a folder it doesn’t immediately pickup changes) and does sync on some specific operations
- Encryption for stored files
- Backup solution with borgbackup and repo for it
- Built very efficient with Rust

- Search and analytics
- WAL (Write-Ahead Logging) to ensure file integrity
- Handles very large files efficiently with concurrent and resumable transfers
- Seamlessly browser app and desktop apps, also mobile apps
- Extensive cilents, libs, CLI, REST and gRPC

How it works

There are several ways to interact with our service:

- **browser app:** we expose a file manager app built for WebAssembly than can manage all operations and transfers
- **local app:** we also have a local file manager app which is very similar to the browser app, in fact they share the same code. The same, will handle all operations and transfers with the service and with other P2P apps. Will expose files with FUSE or other technologies
- **clients:**
  - **CLI:** a command line interface to interact with our service. It can also expose files with FUSE or other technologies
  - **libs:** we have libs in several languages. They can also expose files with FUSE or other technologies
- **API:** we expose a REST API and gRPC that you can use to manage all operations and transfers. It uses WebSockets to notify you about changes

Sync

Between providers

- You setup your providers on our site, you login and grant access to all providers that you need
- Define sync rules, like just Copy from one provider to another or Two-way sync between them
- From this point on we'll handle syncing all changes between providers (no local app needed for this, it's all happening in the cloud)
- We also offer a cloud storage solution, you can setup our service as a provider and use a local app to sync the files. Then you can take full advantage of syncing those files between providers

Local files

- In case you have some local files you want to keep in sync between multiple devices, without any file storage providers, you can use our local app on each device which will handle sync in P2P manner using QUIC. Apps need to be running for this. This is similar to Resilio Sync
- All devices are authenticated using web of trust to make sure are trusted by you

You can mix these 2, for example you could setup a sync between a local folder and a provider. The local app will push changes to our service which will sync them with the provider and the service will push changes from provider back to local app.

Share

Between providers

- You may share a file from your provider with another person, using our service. They will see the file in their provider, both of you can change the file and the changes are kept in sync.
- The service will notify the user. You will be notified back when they accepted and also when they changed something
- On sharing on the same provider if it supports sharing it will use their build-in sharing functionality. On other cases it will copy the file to destination provider and then keep it in sync between the two of you

With external users

- You can share a file from your providers with someone not using our service or any other storage providers. We'll generate a link, we'll email to them, or you can send them, and they can get it from browser, with a torrent client or sync it with local our client
- If they are using our local app you will be notified when they changed something and they will be notified when you make changes

Local files

- You can quickly share a file directly with someone via a link, they can get it from browser, with a torrent client or sync it with our browser or local app
- If both of you are using the local app you can share directly from the app and files will be kept in sync
- The share can be made in 2 ways:
  - **P2P:** in a peer-to-peer manner, your can use our file manager app in browser (uses WebRTC) or local app (uses HTTP, QUIC and BitTorrent), in both cases they need to be running for the user to download the files
  - **Upload to our service:** with the browser app or local app you can first upload the local file to us and user will download it directly from us. The browser app or local app don't need to be running for the download to happen

Receiving files

- Someone might want to send you a file, you setup a Request Files link, send it to them, and we'll handle how the files gets back to you

You can mix between these, for example you can share a local file and the other person who save it on their provider, both files will be kept in sync.

Or you can create a Request Files link based on a provider folder (or local folder) and others can send files to you from their browser, torrent client, local app or from their provider.

Encrypt

- You may want to keep your files encrypted for privacy. Your provider will not have access to the content of your files, nor metadata. You will need the local app or browser app to access the files. It works with local files too
- Encrypted share: it uses PGP to handle encryption. There are 2 options:
  - **share with users on our service:** a session key will be generated then encrypted with destination's user's public key. The encrypted key will be sent along with the file when sharing. If user is adding the share to providers or local app the file will be decryted on demand using the session key. On the disk or on provider it will be kept encrypted all the time
  - **share with external user:** if user is not using our service or downloads the file with browser or torrent, before downloading the file they will need to upload their public key. After that the file is downloaded as encrypted along with the encrypted session key. User can then decrypt the session key with his private key and then decrypt the file, this can be handled with any PGP client like GPG (instructions will be provided on the download page).

Backup

- We're using borg which handles encryption, deduplication and compression. We offer borg repos that can be used to backup data. Subscription is separate from the Sync and Share services.

There are 2 sources of backups:

- **local files:** using the local app you can setup backups schedule for local files and the local app will handle the backup process
- **provider files:** from browser app or local app you can schedule backups from provider files. The service will need to read all files and changes from the provider and will backup on our repo. Everything is handled by the service, you don't need the local app running for this.

When you want to restore some data you can use the local app, you'll select the archive to restore from and it will be mounted in OS from where you can copy the files. You can also use borg CLI or Vorta for GUI if you want, setup will be provided for you in the local app, browser app and on our website.

Features

All sync operations and changes are applied with WAL (Write-Ahead Logging) to ensure file integrity on crash or power loss. That is, any changes are first written to WAL and then

applied to the file. On crash or power loss, the next time the process starts it will apply remaining changes, this repeats until all changes are successfully applied.

MD5, SHA1 hashes are checked at all times for file integrity. After we transfer the files we will compare local hash with remote hash to ensure data integrity.

All transfers are multi-threaded and resumable.

Sync between providers

After you setup the providers and Sync rules the sync process is automatically handled in the cloud. All your files will stay in sync in real-time and you can access them directly on providers solutions (local client or in browser) or on our local app or browser app.

We also offer a cloud storage solution based on S3 for now, you can setup our service as a provider and use the local app to sync the files. Then you can take full advantage of syncing those files between providers.

This is useful when you want to keep files in sync between multiple and/or different providers.

Sync modes

There are 4 modes to sync the files:

1. **Copy:** it will copy new files and changes from source to destination. No deletes or renames will occur. In case the file is changed on destination also (see how this is determined based on **compare-mode** below) it will resolve the conflict based on conflict resolution (see below)
2. **One-way:** like Copy but will also delete and rename files o destination. Conflicts are handled based on conflict resolution
3. **Move:** will move the changed files from source to destination, deleting them from source after transferred. Conflicts are handled based on conflict resolution
4. **Two-way:** will propagate changes in both directions. In case of changes on both sides conflicts are handled based on conflict resolution

compare-mode

Takes a comma-separated list, with the currently supported values being **size, modtime, and checksum**. For example, you could compare size and modtime but not the checksum.

Conflict resolution

If configured first it will try to **auto-merge** (see below). This is disabled by default as it needs to download the remote file to perform the merge. if it doesn't succeed it will keep the content based on **keep-content-mode** and will copy the other file to a new file with suffix in name like **"conflict-<other>-date"** indicating the other identifier and the date when the change was made.

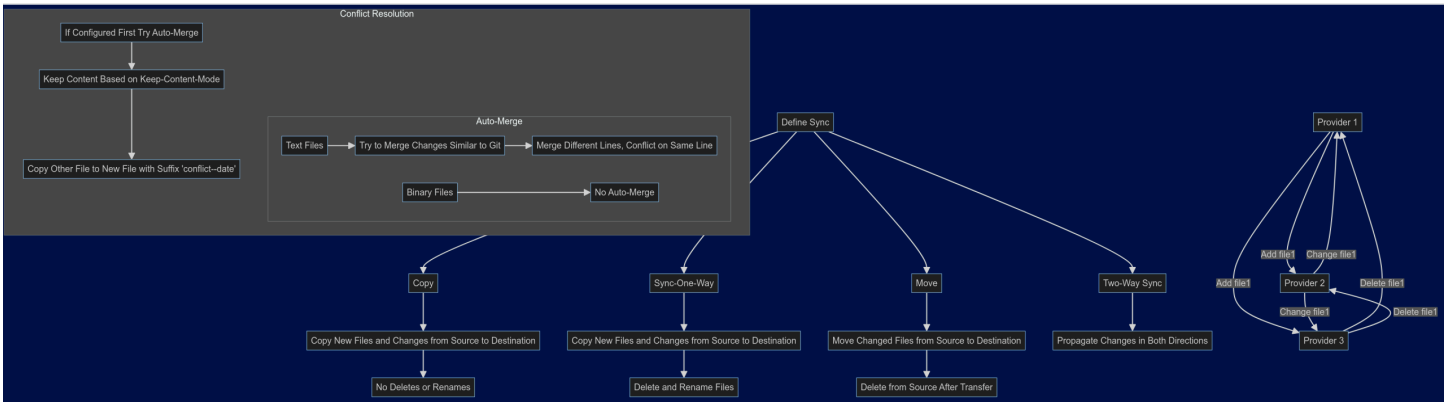
auto-merge:

- **text files:** it will try to merge the changes similar to Git. If there are changes on different lines the files will be merged, if the changes are on the same line it's a conflict and will not merge anything. This is disabled by default as it needs to download the remote file
- **binary files:** it will not do any auto-merge

keep-content-mode

- **newer (default)**: the newer file (by modtime) is considered the winner, regardless of which side it came from. This may result in having a mix of some winners from Path1, and some winners from Path2
- **older**: same as newer, except the older file is considered the winner
- **larger**: the larger file (by size) is considered the winner (regardless of modtime, if any). This can be a useful option for remotes without modtime support, or with the kinds of files (such as logs) that tend to grow but not shrink, over time
- **smaller**: the smaller file (by size) is considered the winner (regardless of modtime, if any)
- **path1**: the version from Path1 is unconditionally considered the winner (regardless of modtime and size, if any). This can be useful if one side is more trusted or up-to-date than the other
- **path2**: same as path1, except the path2 version is considered the winner

If either of the underlying remotes lacks support for the chosen method, it will be ignored and will fall back to the default of newer. If modtime is not supported either by the remote it will fallback to path1.



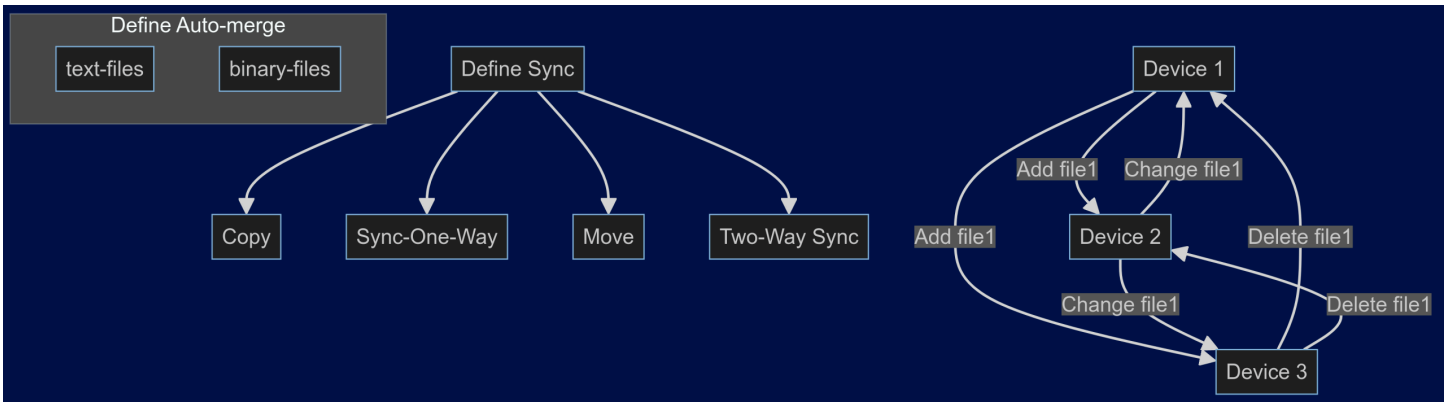
Sync local files

Useful to sync local files between multiple devices with no remote providers involved. It will do it in a P2P manner using QUIC (will fallback to TCP/IP if firewalls blocks UDP). Your local apps need to be running for the sync to happen. This is similar to Resilio Sync.

All devices are authenticated using web of trust to make sure are trusted by you. When you setup a sync an invite link or QR code is be generated. When you open the other app based on these it will send a handshake for the device to join the sync group and you need to confirm it

from any of the other devices in the group. This will give him a cryptographic key synced between all devices and only then other devices will accept sync operations with it.

Sync mode (Copy, Move, One-way, Two-way), compare-mode and conflict resolution are handled as per above.



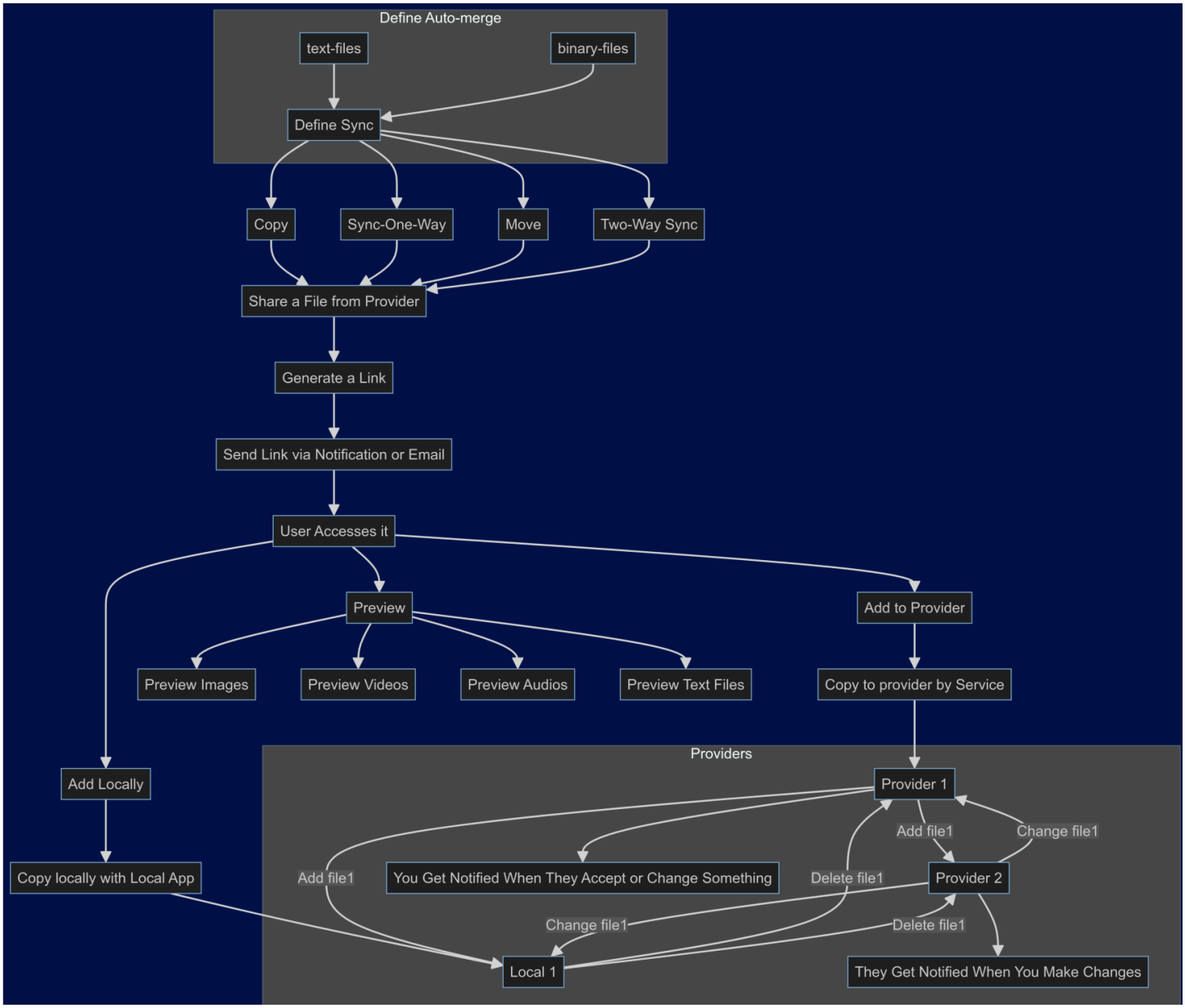
Share between providers

You may share a file from your provider with another person, using our service. They will see the file in their provider, both of you can change the file and the changes are kept in sync. You can also setup read-only shares

On sharing on the same provider if it supports sharing it will use its build-in sharing functionality when possible.

In other cases, it will copy the file to destination provider and then keep it in sync between the two of you. Both of you need to use our service with the providers setup in our service for this to work. Sync mode (Copy, Move, One-way, Two-way), compare-mode and conflict resolution are handled as per above.

The service will notify the user. You will be notified back when they accepted and when they changed something, they will be notified when you changed something.



Share with external users

You can share a file from your providers with someone not using our service or any other storage providers. We'll generate a link, we'll email to them, or you can send them, and they can get it from browser, with a torrent client or sync it with local our client.

You can initiated the share in 2 ways:

- **from browser app:** using our file manager app in browser you will pick-up a file from a provider
- **from local app:** you will initiate the sync from local app

Sync mode (Copy, Move, One-way, Two-way), compare-mode and conflict resolution are handled as per above.

You can define these options when sharing:

- password

- expiry date

You always have the option to stop sharing.

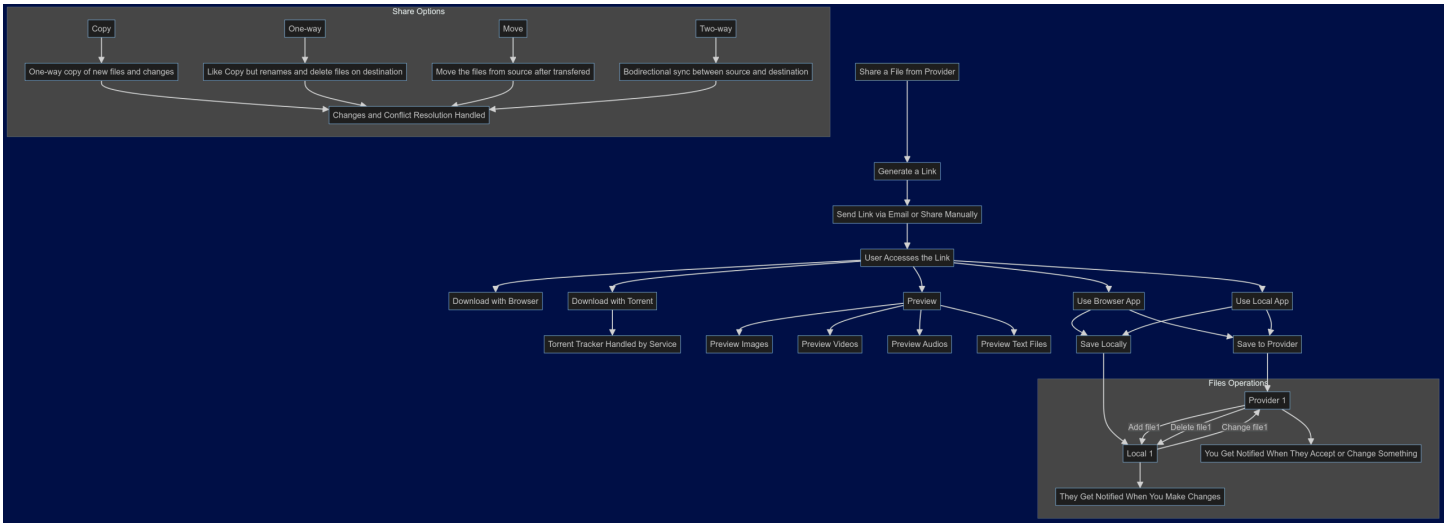
You will be notified when they accepted or change something and they will be notified when you make changes.

When they access the link will have these options:

- **download with browser:** download will be handled by their browser, it's a One-way transfer
- **download with torrent:** they will use a torrent client to get the file. We will handle the torrent tracker. This is a One-way transfer
- **use browser app:** will start the file manager browser app (they can create an account if they want) from where they can add the files to a provider (if they have an account on our service) or save them locally. Files will be kept in sync between the two of you
- **use local app:** they can install and/or use the local app from where they can add the files to a provider (if they have an account on our service) or sync them locally. Files will be kept in sync between the two of you
  - a QR code will also be presented if they want to use the mobile apps
- **preview:** for some types we offer preview, like for images, videos, audios, text files, PDFs, ...

I can handles very large files efficiently with parallel and resumable downloads.

Sync mode (Copy, Move, One-way, Two-way), compare-mode and conflict resolution are handled as per above.



Share Local files

You can quickly share a file directly with someone via a link, they can get it from browser, with a torrent client or sync it with our apps.

If both of you are using any of our apps you can share it directly from the app and files will be kept in sync.

The file transfer can be handled in 2 ways:

- **P2P:** in a peer-to-peer manner, your can use our file manager app in browser (uses WebRTC) or local app (uses HTTP, QUIC and BitTorrent), in both cases the apps need to be running for the user to download the files
- **Upload to our service:** with the browser app or local app you can first upload the local files to us and user will download them directly from us. The browser app or local app don't need to be running for download to happen

Sync mode (Copy, Move, One-way, Two-way), compare-mode and conflict resolution are handled as per above.

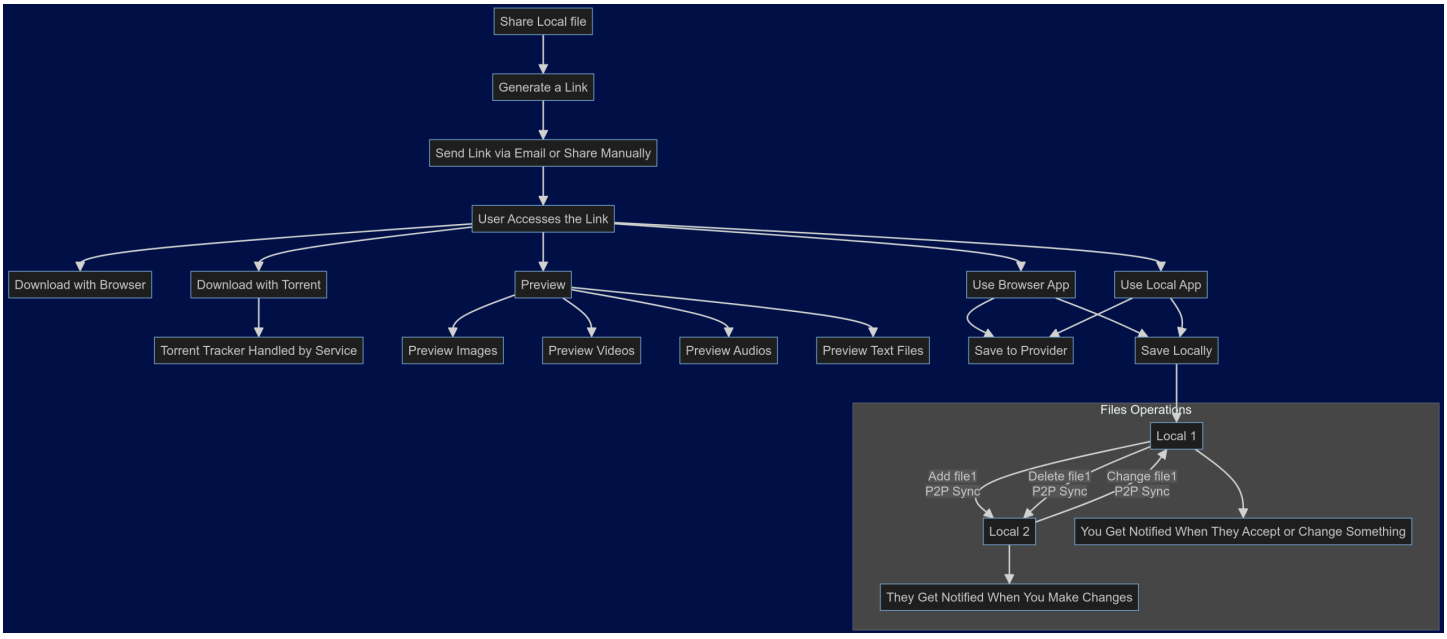
You can define these options when sharing:

- password
- expiry date

You always have the option to stop sharing.

You will be notified when they accepted or change something and they will be notified when you make changes.

When they access the link they have the same options as above for sharing with external users.



Receiving files

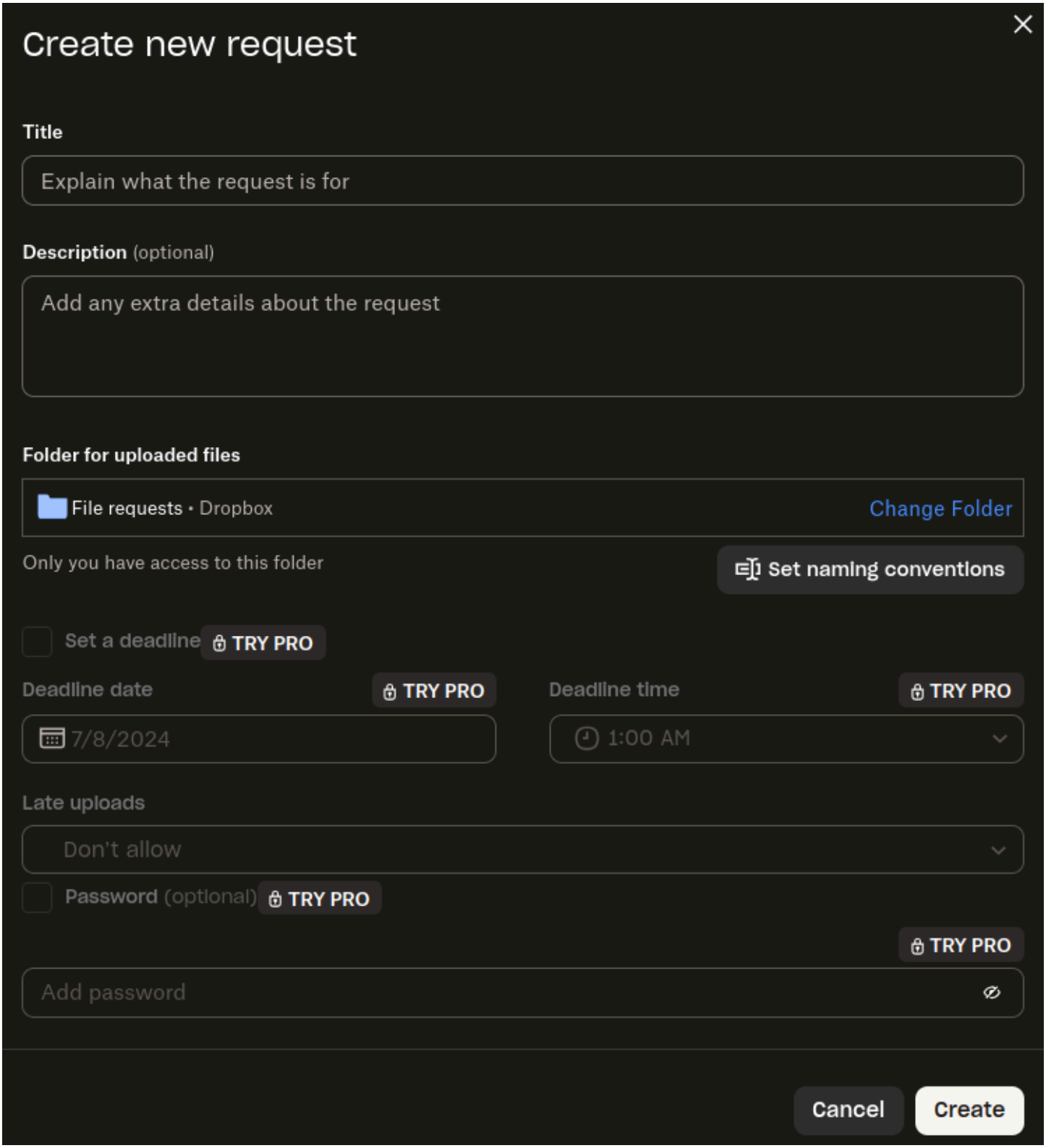
Someone might want to send you a file. For that you first setup a Request Files link, send it to them, and we'll handle how the file gets back to you.

This is similar to S3 presigned URLs but create a dest folder where others can upload more files and even folders.

Useful for One-way transfers only. After upload the files will not be kept in sync anymore.

You can create such an URL from a provider folder or local folder. For local folders the browser app or local app need to be running when the user upload the files.

We'll notify the user with the link, or you can send it to them. You will be notified when they uploaded new files.



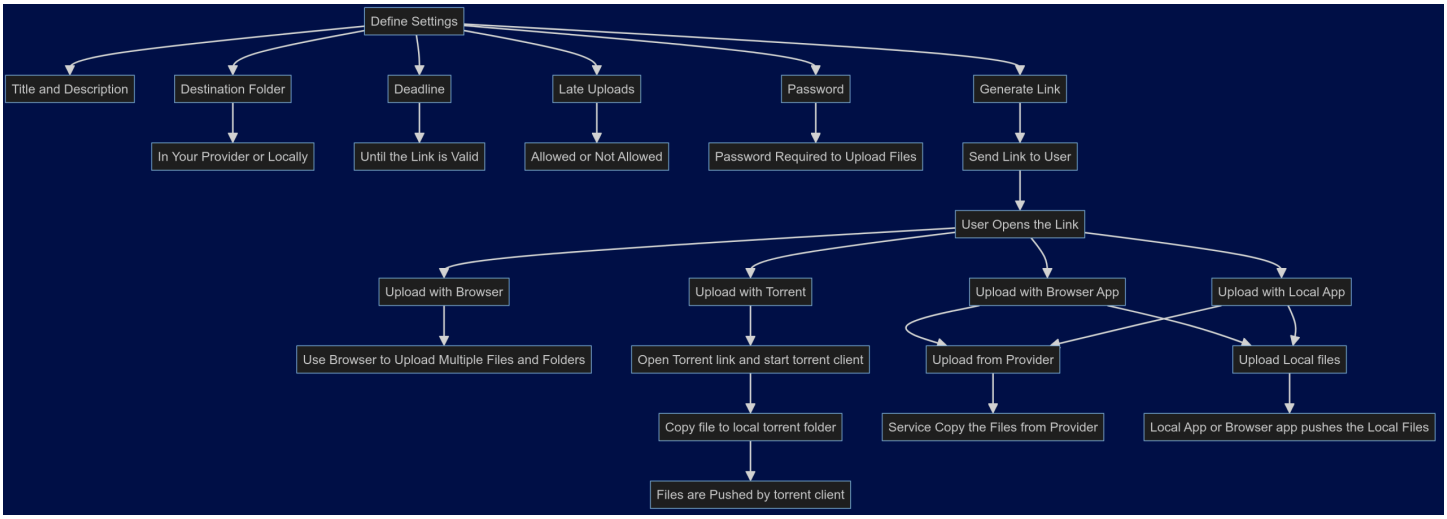
You can define these:

- title and description
- destination folder: in your provider or locally
- deadline: until the link is valid
- late uploads: if users are allowed to upload after this date
- password: a password user need to use to upload files

When user opens the link it will be presented with:

- **upload with browser:** will use the browser to upload multiple files and folders

- **upload with torrent:** based on a link the user's torrent client will be started and they can upload by adding files or folders to their local folder linked to the torrent
- **upload with browser app:** will start the browser file manager app (they can create an account if they want) from where they can choose to upload files from a provider (if they have an account on our service) or local files
- **upload with local app:** they can install and/or use the local app from where they can choose to upload a files from a provider (if they have an account on our service) or local files
    - a QR code will also be presented if they want to use the mobile apps



Encrypt

In transit data (between our apps, service, provider, browser, torrent clients) is always encrypted using TLS 1.3 when possible. But you may decide to keep your files encrypted, for privacy reasons. Your provider will not have access to the content of your files, nor metadata. You will need the local app or browser app to access the files. Works with local files too.

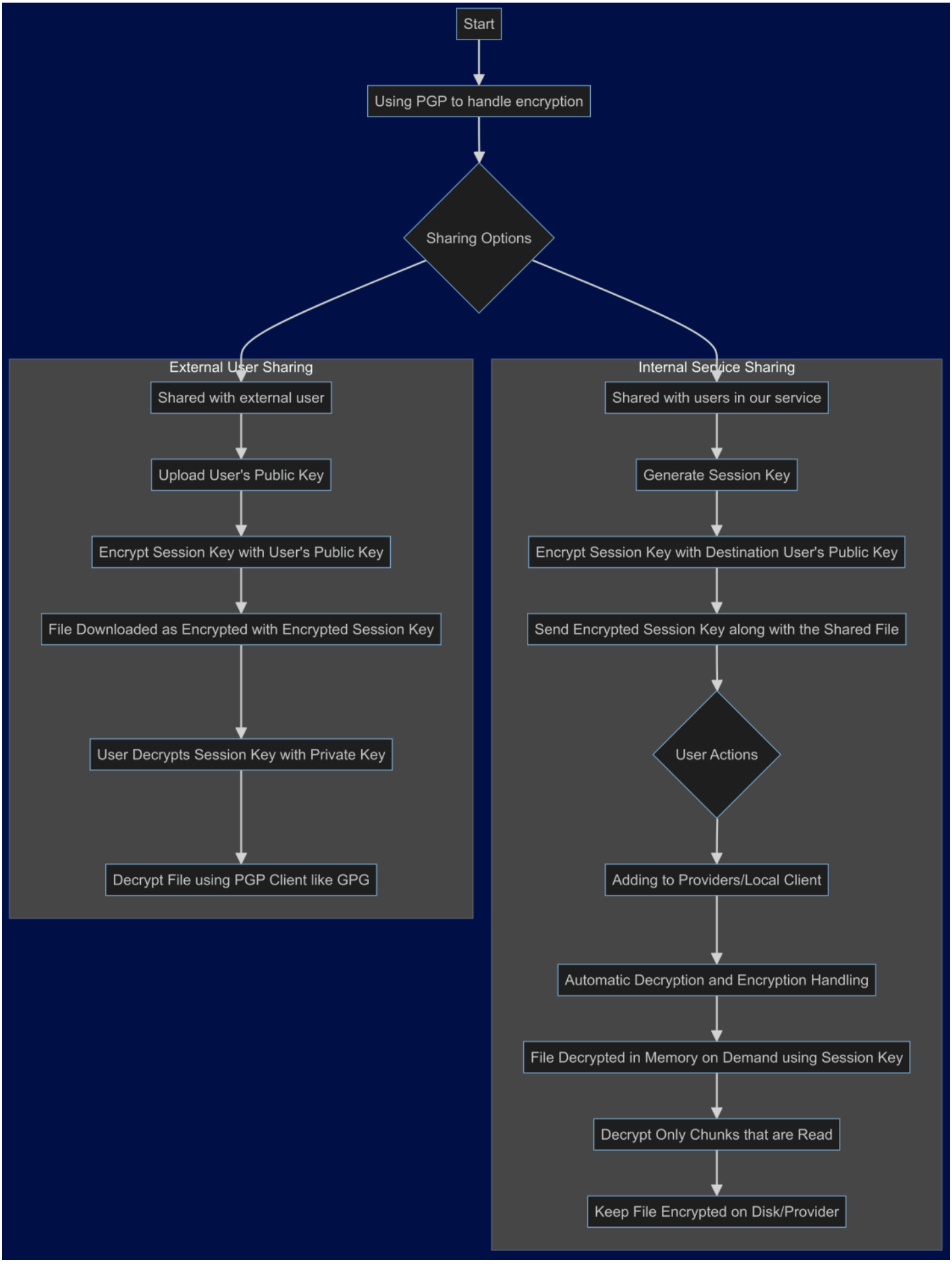
You can choose to encrypt the whole provider content or just selected files or folders.

Encrypted share

It uses PGP to handle encryption. There are 2 options:

- **share with users in our service:** a session key will be generated then encrypted with destination's user public key. The encrypted key will be sent along when sharing. If user is adding the share to providers or local app the decryption and encryption will be handled automatically. The file will be decryted in memory on demand using the session key (decrypting only the chunks that are read). On the disk or in provider it will be kept encrypted all the time
- **share with external users:** if users are not using our service or users download the file with browser or torrent, before they can download they will need to upload their public key. After that the file is downloaded as encrypted along with the encrypted session key. Users can then decrypt the session key with their private key and then

decrypt the file, this can be handled with any PGP client like GPG. If user chooses to use our browser app or local app all this flow will be handled by the app



Browser app built in WebAssembly

We have a browser app as a file manager built in WebAssembly that is compatible with all major browsers. You can perform all operations from it and it handles the traffic also. Concurrent and resumable transfers.

It supports notifications that you receive for several events and changes, like when someone is sharing a file with you, someone accepted your share, someone changed a shared file and others.

Local app for desktop and mobile

We have local apps for all major desktop OSs and on mobile for Android and iOS.

The desktop app is very similar to browser app, in fact they share the same code. The same, will handle all operations and transfers with the service and with other P2P apps. Will expose files with FUSE on Linux and macOS, WinFSP (or others) on Windows and file picker on mobile. Concurrent and resumable transfers.

It operates in 2 modes:

- **encrypted cache (default):** it downloads the files on demand in real-time and keep them in a local encrypted cache with a maximum size
- **full copy kept in sync:** it will download all files for each provider and keeps them in Two-way sync

It supports notifications that you receive for several events and changes, like when someone is sharing a file with you, someone accepted your share, someone changed a shared file and others.

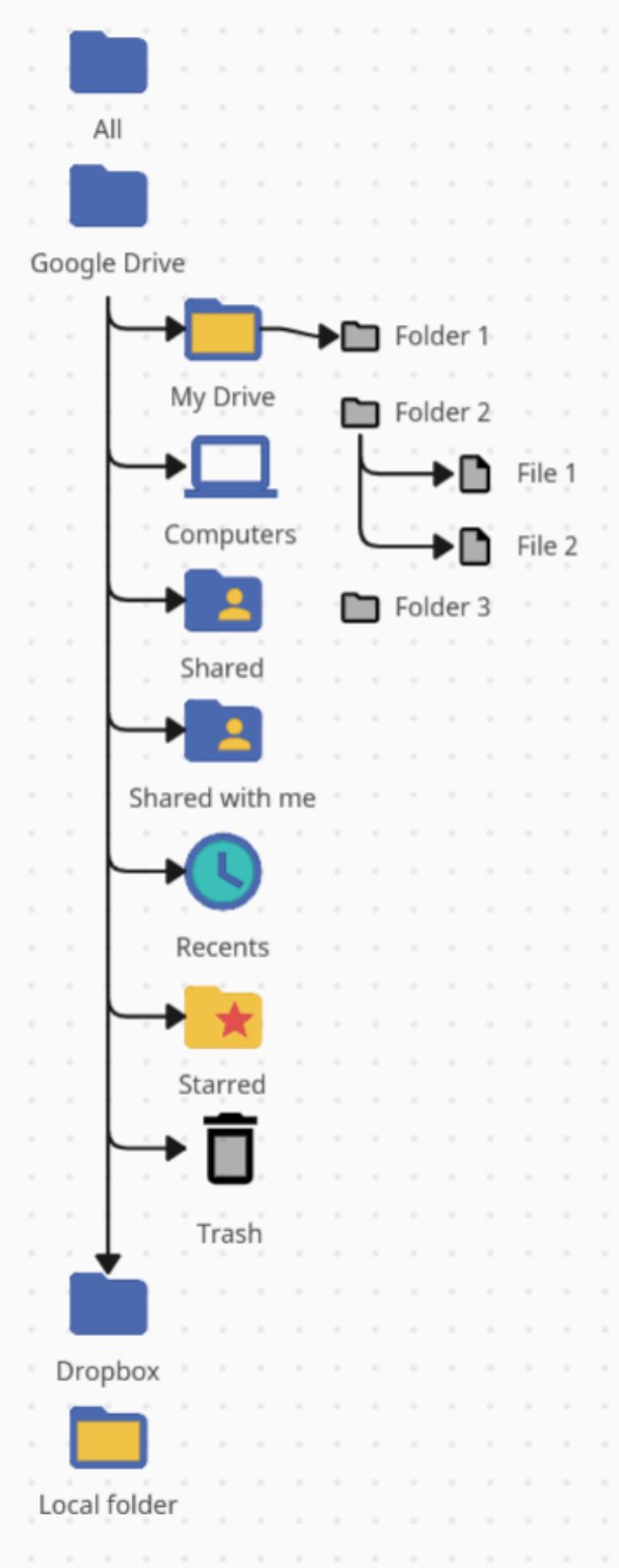
Handles very large files efficiently with concurrent and resumable transfers for the browser app, local app and shared files

All transfers are made concurrently and are resumable. This is true for browser app, local app (if they crash or closed while transferring the next time they star the transfer will continue) and for shared files (browser transfers and with torrent clients).

When possible we use BitTorrent with QUIC to sync files, this is true between local files for ex, where not we use the supported provider protocols.

Local views: My Drive, Computers, Shared, Shared with me, Recents, Starred, Trash

You will have a dedicated entry point folder for each provider and also for local folders. Inside each folder you there will be some basic categories. You can perform all operations on them, create new files, copy between provider or local folders, share, search, ...



Global view from all providers and local files, My Drive, Computers, Shared, Shared with me, Recents, Starred, Trash

Like local views but gives you an overview of files from all providers and local folders accessed based on some basic categories. Corresponds to All folder in the above diagram. You can perform all operations on them, create new files, copy between them, share, search, ...

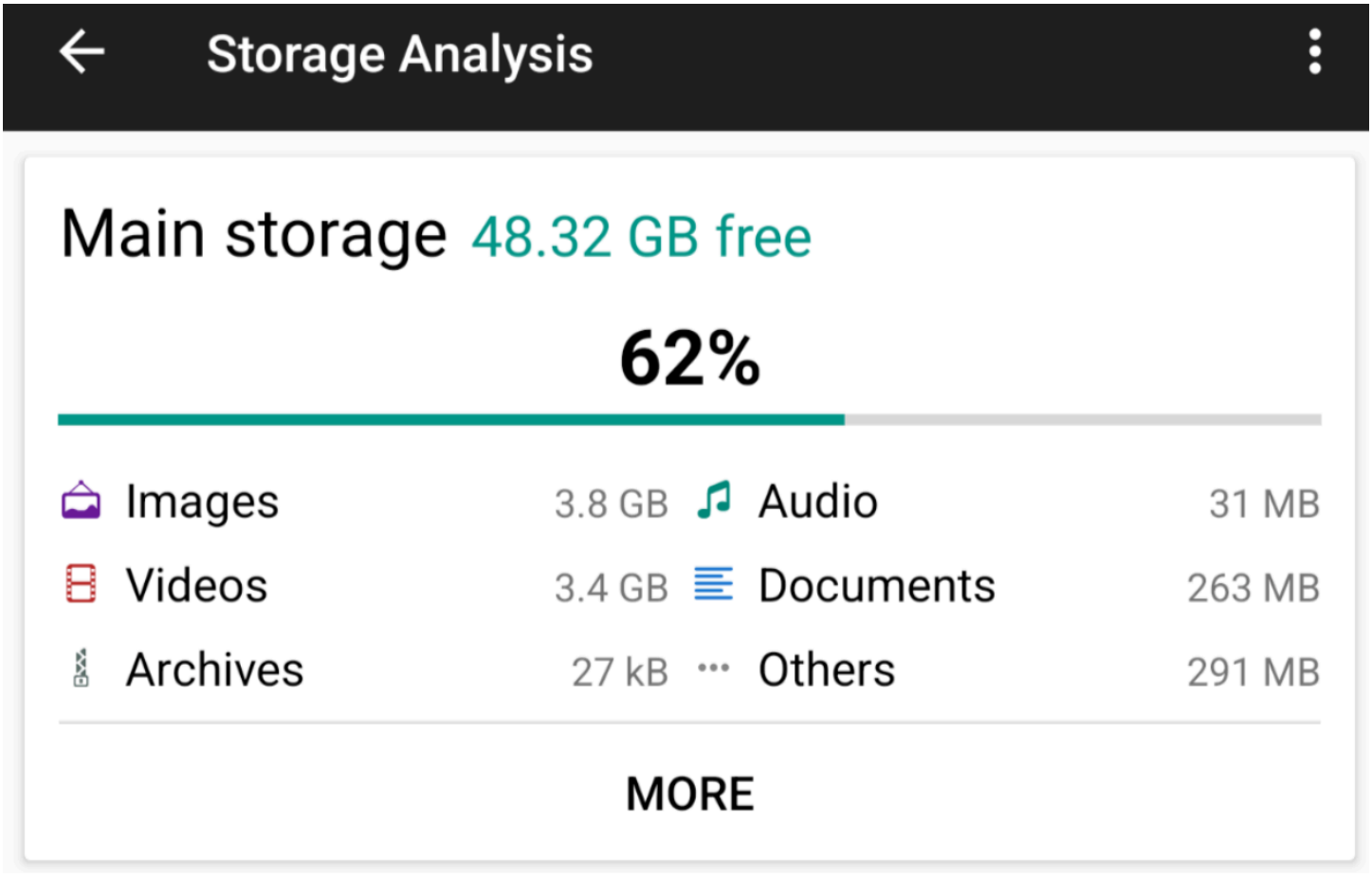
Internal links to other files

Links to files inside same provider or local files. This is normally dependent on the underlying remote support, but in cases it's not supported we do our best to implement inks which then you can only manage and access with our apps.

Search capabilities

- **metadata fields:** like name, size, modified date
- **advanced search:** like shared with, last modified by
- **file content:** search inside file content, this is dependent on the underlying remote support. You might opt-in for us analyzing the files and offer content search capabilities but this would require us to read all files

Analytics



We offer various analytics like:

- stats on types of files, images, videos, ...
- metrics on files metadata, files size, changed date, changed by, ...
- stats per provider, space used, free, transferred MB per month, in total, ...
- metrics on files activity, how often a file is changed, who did the most/recent change, most changed/used files, unused files ...
- duplicate files between providers

File history and versioning

History

Many providers offer a history of changes and operations for each file. Where supported on we will use that, where not we will try to implement and make it available in apps and clients. The same will offer this for local files.

This mostly refers to name change and activity history, not convent history, see versioning for that.

Versioning

Many providers support having multiple versions of the same file, as you copy a new file with same name or change it, it will be a new version. When supported by providers we will use it, when not we will try to implement it our own and make it available in apps and clients. Works also for local files.

Implementation note: this could be a good use case of git repos. Git objects are compressed.

Integration with other systems

We consider providing integration with:

- Google Workspace
- Microsoft
- Slack
- Adobe
- HubSpot
- Autodesk
- Canva
- Figma
- Miro
- AWS

Cleanup storage

Based on the analytics overview you can choose what files to delete when you need to make more space on provider or locally.

Sync, share status, storage overview

We offer some health overview for you like:

- **connection status:** between your local apps, our service and providers
- **sync status:** when was the last sync, if was successful or it failed, pending changes to be synced locally, pending local changes to be synced to providers, ...
- **share:** pending shared files to be accepted, pending Request Files, ...
- **storage stats:** basic stats like used/free space per provider, type of files, ...

Backups with borg, encrypted, deduplicated, compressed

We're using borg which handles encryption, deduplication and compression. We offer borg repos that can be used to backup data. Subscription is separate from the Sync and Share services.

There are 2 scopes of backups:

- **local data:** using the local app or browser app you can setup backup schedule for local files, this will be done with cron jobs, scheduled tasks on Windows, termux-job-scheduler on Android and ibimobiledevice in iOS
- **provider files:** from browser app or local app you can schedule backups from provider files. The service will need to read all files and changes from the provider and will backup on our repo. All the process is handled by the service, you don't need the local app running for this.

When you want to restore some data you can use the local app or browser app, you'll select the archive to restore from, you can then access files in the app and copy them locally or to a provider. In the local app you can also mount it in the OS from where you can copy the files. You can also use borg CLI or Vorta for GUI if you want, setup will be provided for you in the local app, browser app and on our website.

Automation

You will be able to define some automation actions for folders. When files are added or changed in them the following actions could be performed:

- convert to PDF
- convert image
- convert audio/video
- unzip
- watermark files

Photos manager

We will offer a basic browser based and local app photo manager. It can manage photos from all providers, or local photos, offers basic edit operations, share photos, albums, stats,

REST API, gRPC, CLI clients and client libs in multiple languages

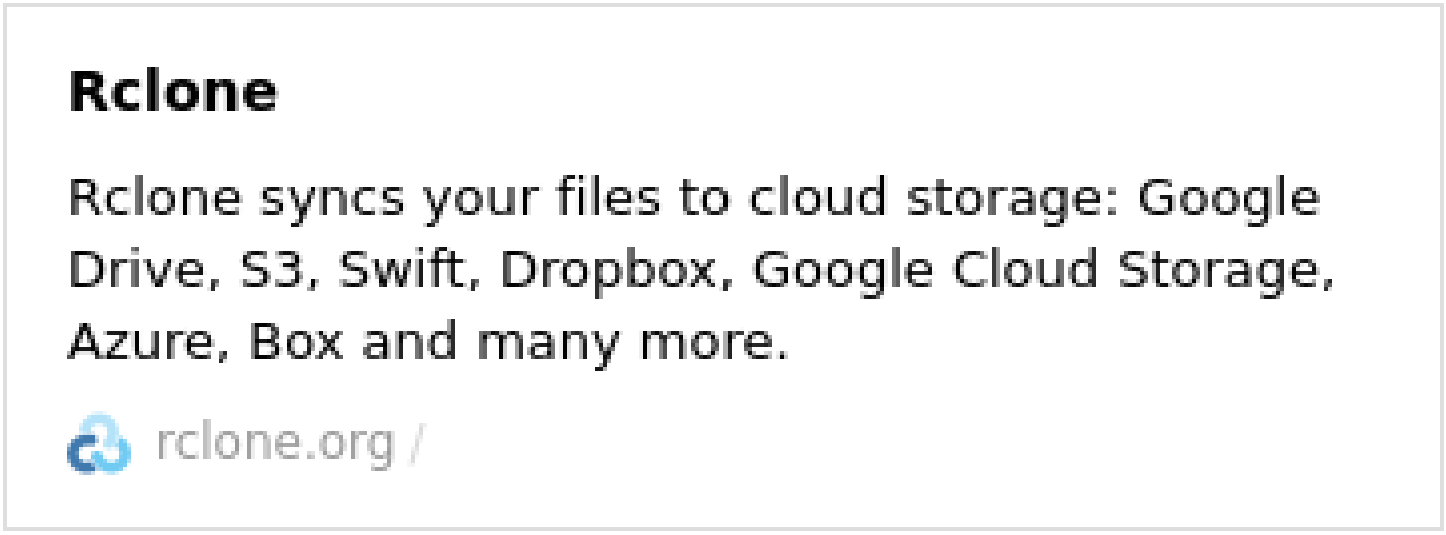
We will expose a REST API and gRPC service and will have client libs in multiple languages that users can use to build their own apps if they want.

Also we will have a CLI that can be used to automate for example various flows.

Many supported providers

Initially we will use rclone on the service side and we will support all their supported providers.

In the future we plan to implement our own clients for several providers which might offer more granular and efficient sync. It helps us for example to catch remote changes almost as they happen, as soon as we are notified by the remote, and propagate the changes to other destinations, without needing to use file watchers or rely on the sync logic of rclone provider implementation.



Stack

We’ll build it mostly in Rust with AWS and Kubernetes.

Scope	Solution
REST API, gRPC	axum, tonic
Websocket	tokio-tungstenite
DB	RDS, ScyllaDB, Google Cloud Spanner, Neo4j
Local DB	SurrealDB
Browser app	egui, wasi, wasm-bindgen, rencfs, pgp
Local app desktop	egui, mainline, transmission_rs, quinn, cratetorrent/rqbit, rencfs, pgp, fuse3
Local app mobile	Kotlin Multiplatform
Sync daemons	tokio, rclone, rencfs
Use Kafka	rdkafka
Keycloak	axum-keycloak-auth (in app token verificaton) or Keycloak Gatekeeper (reverse proxy in front of the services)
Event Bus	Kafka / Amazon SQS / RabbitMQ
File storage	S3
Search and Analytics	ELK, Apache Spark, Apache Flink, Apache Airflow
Identity Provider	Keycloak
Cache	Redis
Deploy	AWS Lambda and Amazon EKS
Monitoring	Prometheus
Logs	Elastic ELK