Gokapi Documentation

Release 1.2.1

Marc Ole Bulling

Aug 17, 2022

CONTENTS

Gokapi is a lightweight server to share files, which expire after a set amount of downloads or days. It is similar to the discontinued Firefox Send, with the difference that only the admin is allowed to upload files.

This enables companies or individuals to share their files very easily and having them removed afterwards, therefore saving disk space and having control over who downloads the file from the server.

Identical files will be deduplicated. An API is available to interact with Gokapi. AWS S3 and Backblaze B2 can be used instead of local storage. Customization is very easy with HTML/CSS knowledge.

CHAPTER

ONE

CONTENTS

1.1 Setup

There are two different ways to setup Gokapi: Either a bare metal approach or docker.

Also there are two different versions: *Stable* indicates that you are using the latest release which should work without any bugs. *Unstable* is the latest developer version, which might include more features, but could also contain bugs.

1.1.1 Installation

Bare Metal

Stable version

Download the project and copy the executable into a new folder with write permissions.

Unstable version

Only recommended if you have expierence with the command line. Go 1.18+ needs to be installed.

Create a new folder and in this folder execute

```
git clone https://github.com/Forceu/Gokapi.git .
go generate ./...
go build Gokapi/cmd/gokapi
```

This will compile the source code and create an executable from the latest code.

Docker

To download, run the following command, and replace YOURTAG with either *latest* (stable) or *latest-dev* (unstable).

```
docker pull f0rc3/gokapi:YOURTAG
```

Most of the time, you will need the *latest* tag.

If you don't want to download the prebuilt image, you can find the Dockerfile on the Github project page.

1.1.2 First Start

After the first start you will be redirected to a setup webpage. To change the port for the setup please set the GOKAPI_PORT env variable, see *Environment variables*

Starting Gokapi

Bare Metal

To start Gokapi, execute the binary with your command line or by double clicking.

Docker

To start the container, run the following command:

```
docker run -v gokapi-data:/app/data -v gokapi-config:/app/config -p 127.0.0.
↪1:53842:53842 f0rc3/gokapi:latest
```

With the argument `-p 127.0.0.1:53842:53842` the service will only be accessible from the machine it is running on. In most usecases you will use a reverse proxy for SSL - if you want to make the service available to other computers in the network without a reverse proxy, replace the argument with `-p 53842:53842`. Please note, unless you select SSL during the setup, the traffic will not be encrypted that way and data like passwords and transferred files can easily be read by third parties!

Initial Setup

During the first start, a new configuration file will be created and you will be asked for several inputs. With your webbrowser open `http://localhost:53842/setup` (or the appropriate URL) and follow the setup.

Webserver

The following configuration can be set:

- **Bind to localhost** Only allow the server to be accessed from the machine it is running on. Select this if you are running Gokapi behind a reverse proxy or for testing purposes
- **Use SSL** Generates a self-signed SSL certificate (which can be replaced with a valid one). Select this if you are not running Gokapi behind a reverse proxy. Please note: Gokapi needs to be restarted in order to renew a certificate.
- **Webserver Port** Set the port that Gokapi can be accessed on
- **Public Facing URL** Enter the URL where users from an external network can use to reach Gokapi. The URL will be used for generating download links
- **Redirection URL** By default Gokapi redirects to this URL instead of showing a generic page if no download link was passed

Authentication

This menu guides you through the authentication setup, where you select how an admin user logs in (only user that can upload files)

Username / Password

The default authentication method. A single admin user will be generated that authenticates with a password

OAuth2 OpenID Connect

Use this to authenticate with an OIDC server, eg. Google, Github or an internal server. *Note:* If a user is revoked on the OIDC server, it might take several days to affect the Gokapi session.

Option	Expected Entry	Example
Provider URL	The URL to connect to the OIDC server	https://accounts.google.com
Client ID	Client ID provided by the OIDC server	[random String]
Client Secret	Client secret provided by the OIDC server	[random String]
Allowed users	List of users that is allowed to log in as an admin. Separate users with a semicolon or leave blank to allow any authenticated user	gokapiuser@gmail.com;companyadmin@gmail.c

When creating an OIDC client on the server, you will need to provide a **redirection URL**. Enter `http[s]://[gokapi URL]/oauth-callback`

You can find a guide on how to create an OIDC client with Github at Setting up GitHub OAuth 2.0 and a guide for Google at Setting up OAuth 2.0.

Header Authentication

Only use this if you are running Gokapi behind a reverse proxy that is capable of authenticating users, e.g. by using Authelia or Authentik.

Enter the key of the header that returns the username. For Authelia this would be `Remote-User` and for Authentik `X-authentik-username`. Separate users with a semicolon or leave blank to allow any authenticated user, e.g. `gokapiuser@gmail.com;companyadmin@gmail.com`

Access Restriction

Only use this if you are running Gokapi behind a reverse proxy that is capable of authenticating users, e.g. by using Authelia or Authentik.

This option disables Gokapis internal authentication completely, except for API calls. The following URLs need to be restricted by the reverse proxy:

- `/admin`
- `/apiDelete`

- `/apiKeys`
- `/apiNew`
- `/delete`
- `/e2eInfo`
- `/e2eSetup`
- `/uploadChunk`
- `/uploadComplete`

Warning: This option has potential to be *very* dangerous, only proceed if you know what you are doing!

Storage

Here you can choose where uploaded files shall be stored. Use the option to always store image files to the local storage, if you want to use encryption for cloudstorage, but require hotlink support.

Local Storage

Stores files locally in the subdirectory `data` by default.

Cloudstorage

Stores files remotely on an S3 compatible server, e.g. Amazon AWS S3 or Backblaze B2. Please note that files will be stored in plain-text, if no encryption is selected later on.

It is highly recommended to create a new bucket for Gokapi and set it to "private", so that no file can be downloaded externally. For each download request Gokapi will create a public URL that is only valid for a couple of seconds, so that the file can be downloaded from the external server directly instead of routing it through the local server.

You then need to create an app key with read-/write-access to this bucket. If you are planning to use the encryption feature, make sure to set the bucket's CORS rules to allow access from the Gokapi URL.

The following data needs to be provided:

Key	Description	Required	Example
Bucket	Name of the bucket in use	yes	gokapi
Region	Name of the region	yes	eu-central-1
KeyId	Name of the API key	yes	keyname123456789
KeySecret	Value of the API key secret	yes	verysecret123
Endpoint	Endpoint to use. Leave blank if using AWS S3.	only for Backblaze B2	s3.eu-central-001.backblazeb2.com

Encryption

Warning: Encryption has not been audited.

There are three different encryption levels, level 1 encrypts only local files and level 2 encrypts local and files stored on cloud storage (e.g. AWS S3). Decryption of files on remote storage is done client-side, for which a 2MB library needs to be downloaded on first visit. End-to-End encryption (level 3) encrypts the files client-side, therefore even if the Gokapi server has been compromised, no data should leak to the attacker.

There are some drawbacks of using encryption:

	No Encryption	Level 1 Local	Level 2 Full	Level 3 End-to-End
File Encryption	None	Only local files	Local and cloud storage	Local and cloud storage
Hotlink Support	Yes	Yes	Only local files	No
Download Progress Indication	Yes	Only cloud storage	No	No
Download Speed	Full	Might be slower for local files	Slower for remote files, might be slower for local files	Slower for all files

You can choose to store the key in the configuration file, which is preferred if access by other parties to your configuration file is unlikely.

If you are concerned that the configuration file can be read, you can also choose to enter a master password on startup. This needs to be entered in the command line however and Gokapi will not be able to start without it.

Please note: If you re-run the setup and enable encryption, unencrypted files will stay unencrypted. If you change any configuration related to encryption, all already encrypted files will be deleted.

1.1.3 Changing Configuration

To change any settings set in the initial setup (e.g. your password or storage location), run Gokapi with the parameter `--reconfigure` and follow the instructions. A random username and password will be generated and displayed in the programm output to access the configuration webpage, as all entered information can be read in plain text (except the user password).

1.2 Usage

1.2.1 Admin Menu

General

After you have started the Gokapi server, you can login using the your admin credentials by going to *http(s)://your.gokapi.url/admin`*

There you can list and manage files and upload new files. You will also see three fields:

- *Allowed downloads* lets you set how many times a file can be downloaded before it gets deleted
- *Expiry in days* lets you set after how many days a file gets deleted latest
- *Password* lets you set a password that a user needs to enter before downloading the file. Please note that the file on the storage server is not encrypted.

Uploading new files

To upload, drag and drop a file, folder or multiple files to the Upload Zone. You can also directly paste an image from the clipboard. If you want to change the default expiry conditions, this has to be done before uploading. For each file an entry in the table will appear with a download link.

Identical files are deduplicated, which means if you upload a file twice, it will only be stored once.

Sharing files

Once you uploaded an file, you will see the options *Copy URL* and *Copy Hotlink*. By clicking on *Copy URL*, you copy the URL for the Download page to your clipboard. A user can then download the file from that page.

If a file does not require client-side decryption, you can also use the *Copy Hotlink* button. The hotlink URL is a direct link to the file and can for example be posted as an image on a forum or on a website. Each view counts as a download. Although Gokapi sets a Header to explicitly disallow caching, some browsers or external caches may still cache the image if they are not complient.

File deletion

Every hour Gokapi runs a cleanup routine which deletes all files from the storage that have been expired. If you click on the *Delete* button in the list, that file will be deleted from the disk immediately. AWS files are deleted after 24 hours, as of right now there is no proper way to find out if a download has been completed.

1.2.2 API Menu

In the API menu you can create API keys, which can be used for API access. Please refer to *API*.

1.3 Updating Gokapi

1.3.1 Docker

To update, run the following command:

```
docker pull f0rc3/gokapi:YOURTAG
```

Then stop the running container and follow the same steps as in SETUP. All userdata will be preserved, as it is saved to the bbuddy volume (-v command)

1.3.2 Bare Metal

Stable version

To update, download the latest release and unzip it to the directory that contains the old version. Overwrite any existing files.

Unstable version

To update, execute the command `git pull` and then rebuild the binary with `go build Gokapi/cmd/gokapi`.

1.4 Advanced usage

1.4.1 Environment variables

Environment variables can be passed to Gokapi - that way you can set it up without any interaction and pass cloud storage credentials without saving them to the filesystem.

Passing environment variables to Gokapi

Docker

Pass the variable with the `-e` argument. Example for setting the username to *admin* and the password to *123456*:

```
docker run -it -e GOKAPI_USERNAME=admin -e GOKAPI_PASSWORD=123456 f0rc3/gokapi:latest
```

Bare Metal

Linux / Unix

For Linux / Unix environments, execute the binary in this format:

```
GOKAPI_USERNAME=admin GOKAPI_PASSWORD=123456 [...] ./Gokapi
```

Windows

For Windows environments, you need to run `setx` first, e.g.:

```
setx GOKAPI_USERNAME admin
setx GOKAPI_PASSWORD 123456
[...]
Gokapi.exe
```

Available environment variables

Name	Action	Persistent*	Default
GOKAPI_CONFIG_DIR	Sets the directory for the config file	No	config
GOKAPI_CONFIG_FILE	Sets the name of the config file	No	config.json
GOKAPI_DATA_DIR	Sets the directory for the data	Yes	data
GOKAPI_LENGTH_ID	Sets the length of the download IDs. Value needs to be 5 or more	Yes	15
GOKAPI_MAX_FILESIZE	Sets the maximum allowed file size in MB	Yes	102400 (100GB)
GOKAPI_MAX_MEMORY_UPLOAD	Sets the amount of RAM in MB that can be allocated for an upload. Any upload with a size greater than that will be written to a temporary file	Yes	20
TMPDIR	Sets the path which contains temporary files	No	Non-Docker: Default OS path Docker: [DATA_DIR]

* Variables that are persistent must be submitted during the first start when Gokapi creates a new config file. They can be omitted afterwards. Non-persistent variables need to be set on every start.

All values that are described in *Cloudstorage* can be passed as environment variables as well. No values are persistent, therefore need to be set on every start.

Name	Action
GOKAPI_AWS_BUCKET	Sets the bucket name
GOKAPI_AWS_REGION	Sets the region name
GOKAPI_AWS_KEY	Sets the API key
GOKAPI_AWS_KEY_SECRET	Sets the API key secret
GOKAPI_AWS_ENDPOINT	Sets the endpoint

1.4.2 API

Gokapi offers an API that can be reached at `http(s)://your.gokapi.url/api/`. You can find the current documentation with an overview of all API functions and examples at `http(s)://your.gokapi.url/apidocumentation/`.

Interacting with the API

All API calls will need an API key as authentication or a valid admin session cookie. An API key can be generated in the web UI in the menu "API". The API key needs to be passed as a header.

Example: Getting a list of all stored files with curl

```
curl -X GET "https://your.gokapi.url/api/files/list" -H "accept: application/json" -H
↪"apikey: secret"
```

Some calls expect parameters as form/post parameter, others as headers. Please refer to the current API documentation.

Example: Uploading a file

```
curl -X POST "https://your.gokapi.url/api/files/add" -H "accept: application/json" -H
↪"apikey: secret" -H "Content-Type: multipart/form-data" -F "allowedDownloads=1" -F
↪"expiryDays=5" -F "password=" -F "file=@yourfile.dat"
```

Example: Deleting a file

```
curl -X DELETE "https://your.gokapi.url/api/files/delete" -H "accept: */*" -H "id:␣
↪PFnh2DlQRS2PVKM" -H "apikey: secret"
```

1.4.3 Customising

By default, all files are included in the executable. If you want to change the layout (e.g. add your company logo or change the app name etc.), follow these steps:

1. Download the source code for the Gokapi version you are using. It is either attached to the specific release on Github or you can clone the repository and checkout the tag for the specific version.
2. Copy either the folder `static`, `templates` or both from the `internal/webserver/web` folder to the directory where the executable is located
3. Make changes to the folders. `static` contains images, CSS files and JavaScript. `templates` contains the HTML code.
4. Restart the server. If the folders exist, the server will use the local files instead of the embedded files
5. (Optional) To embed the files permanently, copy the modified files back to the original folders and recompiled with `go build Gokapi/cmd/gokapi`.

1.5 Contributions

All contributions are very welcome! If you have an issue or would like to add a pull request, please visit our Github page:

https://github.com/Forceu/gokapi

1.6 Changelog

1.6.1 Overview of all Changes

v1.6.1: 17 Aug 2022

- Fixed setup throwing error 500 on docker installation

v1.6.0: 17 Aug 2022

- Use chunked uploads instead of single upload #68
- Add end-to-end encryption #71
- Fixed hotlink not being generated for uploads through API with unlimited storage time
- Added arm64 to Docker latest image
- Added API call to duplicate existing files
- Fixed bug where encrypted files could not be downloaded after rerunning setup
- Port selection is now disabled when running setup with docker
- Added timeout for AWS if endpoint is invalid
- Added flag to disable CORS check on startup
- Service worker for insecure connections is now hosted on Github
- "Noaws" version is not included as binary build anymore, but can be generated manually

v1.5.2: 08 Jun 2022

- Added ARMv8 (ARM64) to Docker image
- Added option to always store images locally in order to support hotlink for encrypted files
- Fixed crash when remote files exist but system was changed to local files after running –reconfigure
- Added warning if incorrect CORS setting are set for AWS bucket
- Added button in setup to test AWS credentials
- Added more build infos to –version output
- Added download counter
- Added flags for port, config and data location, better flag usage overview
- Fixed that a file was reuploaded to AWS, even if it already existed
- Fixed error image for hotlinks not displaying if nosniff is enforced
- Fixed that two text files were created when pasting text
- Fixed docker image in documentation @emanuelduss

v1.5.1: 10 Mar 2022

- Fixed that selection of remote storage was not available during intitial setup
- Fixed that "bind to localhost" could be selected on docker image during initial setup
- Fixed that with Level 1 encryption remote files were encrypted as well
- If Gokapi is hosted under a https URL, the serviceworker for remote decryption is now included, which fixes that Firefox users with restrictive settings could not download encrypted files from remote storage
- Design improvements by @mraif13

v1.5.0: 08 Mar 2022

- Minimum version for upgrading is 1.3
- Encryption support for local and remote files
- Additional authentication methods: Header-Auth, OIDC and Reverse Proxy
- Option to allow unlimited downloads of files
- The configuration file has been partly replaced with a database. After the first start, the configuration file may be read-only
- A web-based setup instead of command line

v1.3.1: 03 Jul 2021

- Default upload limit is now 100GB and can be changed with environment variables on first start
- Fixed upload not working when using suburl on webserver for Gokapi
- Added log file
- Minor performance increase

v1.3.0: 17 May 2021

- Added cloudstorage support (AWS S3 / Backblaze B2)
- After changing password, all sessions will be logged out
- Fixed terminal input on Windows
- Added SSL support
- Documentation now hosted on ReadTheDocs

v1.2.0: 07 May 2021

- Fixed Docker images
- Added API
- Added header to prevent caching by browser / proxy
- Fixed upload timeout
- Added timeouts for server
- Added header to show download progress
- Prevent data races
- Cleanup routine does not delete files anymore while they are being downloaded
- Fixed that env `LENGTH_ID` was being ignored
- Show message if docker container is run on initial setup without `-it`
- A lot of refactoring and minor improvements / bug fixes

v1.1.3: 07 Apr 2021

- Fixed bug where salts were not used anymore for password hashing
- Added hotlinking for image files
- Added logout button

v1.1.2: 03 Apr 2021

- Added support for env variables, major refactoring
- Configurations like length of the ID or salts can be changed with env variables now
- Fixed minor bugs, minor enhancements

v1.1.0: 18 Mar 2021

- Added option to password protect uploads
- Added ability to paste images into admin upload

v1.0.1: 12 Mar 2021

- Increased security of generated download IDs

v1.0: 12 Mar 2021

- First stable release of the program

1.6.2 Upgrading

Upgrading to 1.5

- You need to update to Gokapi 1.3 before updating to Gokapi 1.5
- After the upgrade the config file can be read-only
- Initial setup has to be done through a web interface now, setting Gokapi up through env variables is not possible anymore
- If you would like to use new features like a different authentication method, please run Gokapi with the paramter `--reconfigure` to open the setup
- If you set the length of the file ID to 80 or more, you need to delete all files before running this update

Upgrading to 1.3

- If you would like to use native SSL, please pass the environment variable `GOKAPI_USE_SSL` on first start after the update or manually edit the configuration file
- AWS S3 and Backblaze B2 can now be used instead of local storage! Please refer to the documentation on how to set it up.