For our QR issue of actually encoding larger size files, there are a lot of ideas  
  
In fact a lot of work has already been done on this!  
  
To kick things off, I stumbled upon this *amazing* project by Adab Zaini.

<https://github.com/ADIBzTER/sementara>

**Sementara**

It’s a file-sharing system where QR codes take center stage, and let me tell you, it’s nothing short of genius. Here's the gist of what I learned:

* **The sender** uploads a file to the cloud (in their case, Google Cloud Storage).
* The system spits out a **QR code** and a **download link**.
* **The receiver** can either scan the QR or use the link to grab the file.

What’s super cool is that the QR doesn’t just carry a boring old URL. It’s got a whole JSON structure in there, so it can pack additional data like user IDs, folder info, and more. Oh, and they use fancy tech like WebSocket for real-time communication and plan to bring in **WebRTC** for direct device-to-device sharing.

Adab Zaini’s project shows how to beat QR size limits by uploading large files to **Google Cloud Storage** and encoding their URLs into QRs. It uses **Express.js** for the backend, **Multer** for uploads, **WebSocket** for real-time magic, and a sleek **React** frontend—perfect for making big files QR-ready.

He has briefly discussed about his project in his youtube channel  
  
<https://www.youtube.com/watch?v=ITv152f_F8E>

Now, I came across a couple of neat approaches that really got me thinking. One involves AWS S3 buckets, and the other leverages Google Cloud Storage (GCS). Both are powerful cloud storage solutions, and the best part? These methods are super adaptable—whether you're automating uploads or just transferring massive files with ease.

**Using AWS S3 Buckets**

I used this medium article by D M J

https://medium.com/@DaalA/building-a-cloud-uploader-cli-using--scripting-423096621db3

Here’s how you can roll with AWS S3:

1. **Setup**: You start by installing and configuring the AWS CLI. Just run aws configure and input your credentials. Easy-peasy.
2. **Create Buckets**: Use aws s3 mb to create a bucket for your files. Add permissions with a bucket policy for public access (don’t forget the security part, though!).
3. **Upload via Script**: Write a script that checks if the file exists and then uploads it to your bucket using aws s3 cp. The script also gives feedback—success, failure, or if something’s missing.

For example, run:

./clouduploader.sh myfile.txt my-cool-bucket

Boom! Your file is in the cloud, and you’re ready to grab that public URL for QR encoding.

**Using Google Cloud Storage (GCS)**

If AWS isn’t your jam, GCS is just as awesome:

1. **Authenticate**: Use gcloud auth login to link your terminal with your Google Cloud account.
2. **Install Tools**: Set up the gsutil command-line tool (it’s like the Swiss Army knife for GCS).
3. **Upload Files**: A single command—gsutil cp {local\_file\_path} gs://{bucket\_name}—gets your file into the bucket. Want to upload a whole folder? No problem. Just point gsutil to the folder, and it’ll do the rest.

The cool thing about both methods? Once the file is uploaded, you get a sharable URL. That’s the magic key we can plug into a QR code generator to make sharing large files a breeze!

The GCS usage was done amazingly in the blog by Neaton Ang (You will need medium subscription 😅……..However u can still make out the steps)

https://medium.com/@neatonang/uploading-files-to-google-cloud-storage-gcs-bucket-via-cli-e14cc26bce9b

**Why This is Perfect for Us**

These CLI-based methods not only let us handle large file uploads easily but also keep things scriptable and automatable (ideal for devs like us). Whether it’s AWS, GCS, or some other cloud service, the process remains simple:

1. Upload the file.
2. Get the URL.
3. Encode it into a QR.

Above, we explored solutions that require authentication for functionality. However, not every scenario demands user credentials. Let’s now look into some tools and methods that operate without requiring authentication. These options provide seamless access and usability for specific tasks—perfect for integrating into workflows like encoding large files into QR codes for easier sharing.

**upload2cloud – Cloud Disk Uploader**

The entire implementation was explained in the github page by hyphop where he explains the webdav linux console client thoroughly

https://github.com/hyphop/upload2cloud?utm\_source=chatgpt.com

When handling large files that need to be encoded into QR formats, the first challenge is storing and accessing these files conveniently. upload2cloud is a perfect solution for this use case, offering quick and lightweight uploads to WebDAV-compatible cloud storage services without requiring a GUI or extensive setup.

**Key Features**

* **Lightweight and Fast**: A single 5 KB file does the job.
* **Cross-Compatible**: Works with a wide range of WebDAV-enabled services.
* **Automation-Friendly**: Designed for terminal-based automation, making it easy to integrate into encoding workflows.
* **Free and Open Source**: Modify the script to meet your exact requirements.

**Use Case for QR Encoding**

Upload large files to cloud storage, then generate QR codes containing the file's public link. This ensures quick and seamless sharing without overwhelming the QR code's data capacity.

**Quick Installation & Usage**

Clone the repository:  
  
  
git clone https://github.com/hyphop/upload2cloud

Install the script:  
  
  
wget https://raw.githubusercontent.com/hyphop/upload2cloud/master/dl/upload2cloud && chmod 0777 upload2\*

Use it with your preferred provider:  
  
  
ln -s upload2cloud upload2cloud-yandex

./upload2cloud-yandex [ FILE ]

1. Retrieve the file URL from the provider and encode it into a QR code.

**Example Workflow**

* Upload a file to Yandex Disk using upload2cloud.
* Retrieve the WebDAV public link (e.g., https://webdav.yandex.com/file).
* Use a QR code generator to encode the link for sharing.

**JSCAPE MFT Server – Enabling Anonymous File Uploads**

The official JScrape documentation is below  
  
https://www.jscape.com/blog/how-to-enable-anonymous-file-uploads-on-your-web-file-transfer-server

If you’re working with external collaborators or users who need to contribute large files for encoding, JSCAPE’s “drop zones” provide a secure way to receive uploads without requiring authentication.

**How It Works**

Drop zones enable anonymous users to upload files through a unique URL. These files are stored in a designated directory for easy retrieval. Once uploaded, the file URL can be encoded into a QR code for distribution.

**Steps to Integrate into QR Encoding**

1. Enable a drop zone on JSCAPE MFT Server:
   * Configure the drop zone path and assign it to a registered user (e.g., user1).
   * Obtain the unique URL for the drop zone (e.g., hostname/dropzone/random\_string).
2. Share the drop zone link with collaborators.
3. After receiving the files, retrieve the uploaded file path and encode it as a QR code.

**Advantages for Encoding Workflows**

* Simplifies the process of collecting large files from external sources.
* Ensures organized storage with minimal manual intervention.
* Allows encoding URLs immediately after files are uploaded.

**Anonymous FTP Uploads**

For a more traditional approach, anonymous FTP can be used to share large files for encoding purposes. While less common today, it remains useful for public-facing file repositories.

**Key Features**

* **Ease of Use**: Users can upload files without creating accounts.
* **Direct Integration**: Combine with QR workflows by encoding the FTP file paths.

**Workflow for QR Encoding**

1. Set up an FTP server to allow anonymous uploads to a specific directory.
2. Provide the server’s hostname and upload directory to collaborators.
3. Once files are uploaded, generate their FTP links (e.g., ftp://hostname/pub/filename).
4. Encode these links into QR codes for easy access.

**Important Notes**

* Restrict access to specific directories to ensure security.
* Monitor uploads to prevent misuse or excessive storage usage.

While simple, this approach is ideal for temporary file sharing and quick QR code generation for public downloads.

I referred to Rahul Awati’s article on this topic  
<https://www.techtarget.com/whatis/definition/anonymous-FTP-File-Transfer-Protocol>

There are actually a lot of jugaadu and creative methods to address the task. Ranging rom using the IPFPS technology most commonly used in NFTs and blockchain to creatively using google drive folders to use them as public cloud storages  
  
Ok let us now go through some of these

**1. Publicly Writable Google Drive Folder**

* **How It Works:**Create a Google Drive folder and change its sharing settings to "Anyone with the link can edit." Share this link in the QR code.
* **Steps:**
  1. Create a folder in Google Drive.
  2. Set folder permissions:
     + Right-click the folder → "Share" → Change "General access" to "Anyone with the link" → "Editor."
  3. Generate the folder link.
  4. Upload files via drag-and-drop. Users can directly add files to this folder.
* **URL Generation:**The URL is the link of the shared folder.

**2. Using Temporary File Hosting Services**

* **How It Works:**Services likehttps://upload.com/ orhttps://www.file.io/ let you upload files and generate a temporary URL without authentication.
* **Steps:**

Use the terminal to upload a file:  
  
  
curl -T your\_file.txt https://upload.com/

1. A unique URL is returned after the upload.

* **URL Generation:**The service provides the URL immediately upon upload.

**3. Email-to-Cloud Services**

* **How It Works:**Use email-based file hosting services likehttps://send-anywhere.com/ or https://www.filemail.com/. Files are sent to an email ID and hosted temporarily.
* **Steps:**
  1. Email the file to the service's address.
  2. Receive a link to the uploaded file.
* **URL Generation:**The link is sent to your email after uploading.

**4. Create a Public FTP Server**

* **How It Works:**Host an FTP server on a cheap VPS or personal computer, allowing users to upload files anonymously.
* **Steps:**
  1. Set up an FTP server using tools like FileZilla or vsftpd.
  2. Configure the server for anonymous uploads (read/write permissions).
  3. Share the FTP server's URL in the QR code.
* **URL Generation:**Share the server address with a pre-configured directory path.

**5. Use GitHub Gists for Small Files**

* **How It Works:**Use GitHub Gists to store text-based files like configurations, scripts, or logs.
* **Steps:**

Use GitHub CLI to create a gist:  
  
  
echo "Your content" | gh gist create -

1. The command returns a unique gist URL.

* **URL Generation:**Use the URL generated by the GitHub API.

**6. Decentralized Storage with IPFS (InterPlanetary File System)**

* **How It Works:**IPFS allows you to upload files and generate a unique hash that can be encoded into a QR.
* **Steps:**
  1. Install IPFS locally or use a public gateway likehttps://infura.io/.

Add your file to IPFS:  
  
  
ipfs add your\_file.txt

1. A unique content hash is generated (e.g., Qm...).

**URL Generation:**Use a public IPFS gateway to share the file:  
  
  
https://ipfs.io/ipfs/<hash>

**7. Use Telegram Bots for Hosting**

* **How It Works:**Create a Telegram bot that accepts files and uploads them to its storage. Use the file's link to share.
* **Steps:**
  1. Create a Telegram bot via BotFather.
  2. Use a bot API to upload files and retrieve their direct links.
* **URL Generation:**The bot provides a URL for each uploaded file.

**8. Leverage Pastebin for Text-Based Content**

* **How It Works:**For text-heavy files, you can use Pastebin's API to upload and retrieve a public URL.
* **Steps:**

Use the Pastebin CLI tool or API.  
  
  
curl -X POST -d "api\_option=paste&api\_paste\_code=YourContent" https://pastebin.com/api/api\_post.php

1. The API returns a public URL.

* **URL Generation:**The URL provided by Pastebin after the upload.

**9. Community Cloud Services (Open-Source Alternatives)**

* **How It Works:**Use open-source solutions like [Nextcloud](https://nextcloud.com/) or [OwnCloud](https://owncloud.com/) hosted on a personal server or shared infrastructure.
* **Steps:**
  1. Set up a Nextcloud/OwnCloud instance.
  2. Share public upload links with folder-specific permissions.
* **URL Generation:**URLs are generated dynamically per folder.

**10. Use Free CDN Services (Cloudflare Workers)**

* **How It Works:**Host files on a free CDN like Cloudflare Workers, which provides temporary public URLs.
* **Steps:**
  1. Set up a Cloudflare Worker for file hosting.
  2. Upload files via the worker's upload endpoint.
* **URL Generation:**Use the worker's public endpoint.

**Combining QR Encoding with Creative Hosting**

Once the URL is generated using any of the above methods, you can encode it into a QR code using tools like:

**CLI-based Tool:**  
  
qrencode -o output.png "https://your-upload-url"

* **Web-Based Generators:** Use sites likehttps://www.qr-code-generator.com/.