Photo Uploader app : [Link](https://medium.com/@jgefroh/software-architecture-image-uploading-67997101a034)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

select photo(type) -> upload photo -> server(capacity)

photo - type, id, name, size - valid/invalid type, size should be less then 10 mb

Scaling: consider server timeout & crashing server. suppose trying to upload 10mb image then

1. 10mb should allocated to server

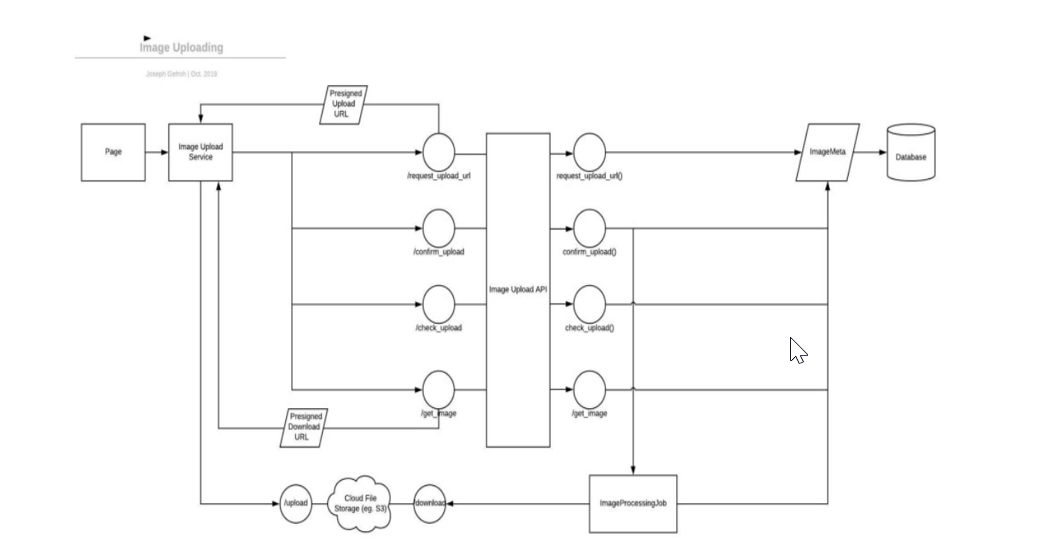
2. a request handler should be associated all the time the image getting uploaded

3. CPU usages to deal with image upload

Security: secure way

Authorization : not all images should be publicly available

ARCHITECTURE :



Signed URLs : is a url having authorization parameter in it.We can use tis url for uploading & downloading images.

Signed Upload URLs

In a simple case,/images/get\_upload\_url could return an unsigned URL: /uploads.

Ex: /images/get\_upload\_url could return a signed URL:  
/uploads?write\_token=a99Xioajksf23.

**Signed Download URLs**If we didn’t have signed download URLs, anyone could download the image

We can gate access behind our own endpoint.  
images/get\_download\_url, which returns a temporary token such as:  
/downloads?read\_token=a99Xioajksf23.

**Cloud File Storage**

We use a separate file storage service (such as S3) to reduce the burden on our web servers.

## Image Metadata Record

We store the metadata of the image record in our database because we have to track it. There’s no sense uploading an image without a way to retrieve or manage it later.

It’s important to note that we do not store the actual image itself in our database, we merely store the metadata, which we then use later to construct the various URLs to it.

Use case :

1. Uploading the image to front end

2. Authorizing user to download image

3. Authorizing user to upload image

4. Uploading image to your server in scalable way

5. Validating image data

6. processing the image, performing cropping, optimization & other tasks

7. Creating image thumbnail, banner

8. storing the image

steps:

1. client request an upload url from the server

2. client upload the image data to upload url

3. client tells server upload is completed

4. server process the image in background

5. client check the image processing status

6. server is done processing image , notifies client

` ```````````````````````````````` X