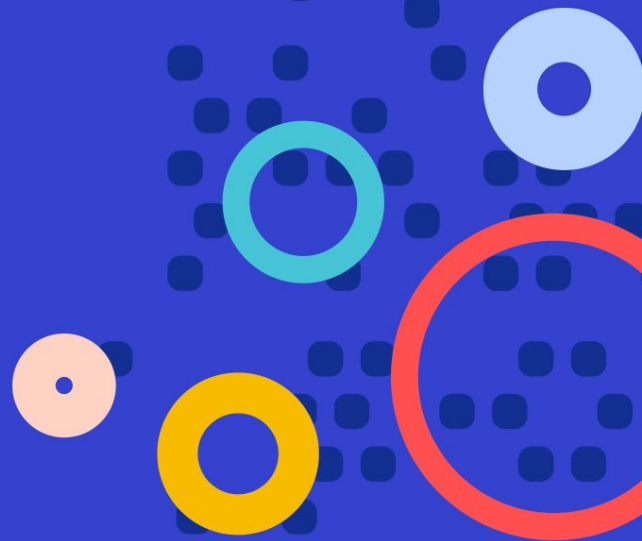


Using Webhooks to Track Asynchronous Processes with Cloudinary

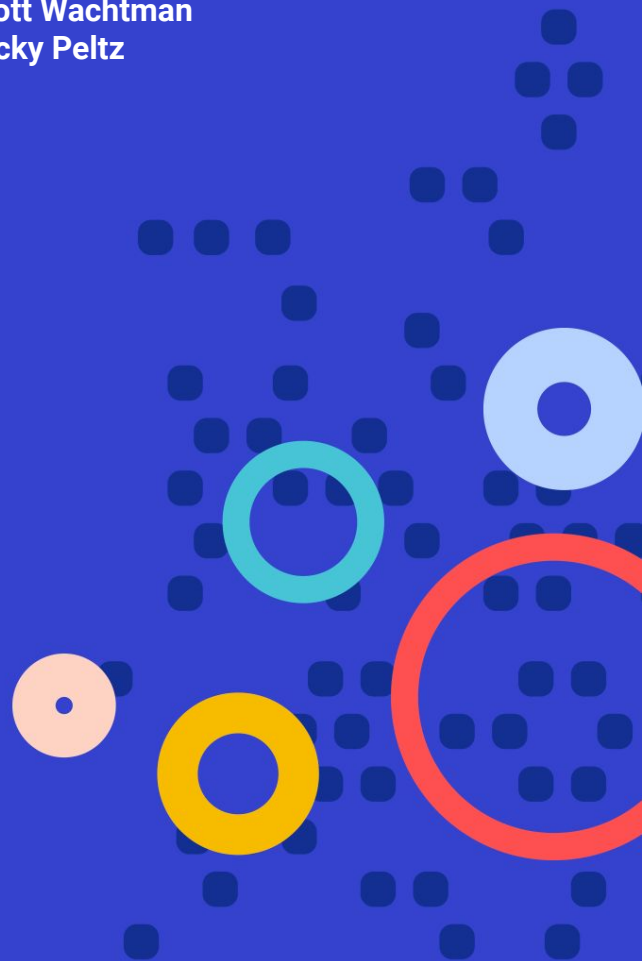
Cloudinary Customer Education



Using Webhooks to Track Asynchronous Processes with Cloudinary

Cloudinary Customer Education, Americas

March 15, 2022





Course Objectives

- **Webhooks:** What are they? Why are they important?
- **Tracking from the DAM:** Media Library Tracking
- **Tracking Upload API Actions:** Cloudinary API Webhook Examples
- **Common Workflow Use Cases:** PIM Update, Upload Widget, Moderation
- **Custom Webhooks**
 - **Notification:** Send Email from a Webhook
 - **Flows:** Video Moderation and Background Removal
- **Security** Securing your Webhook API

What are Webhooks?

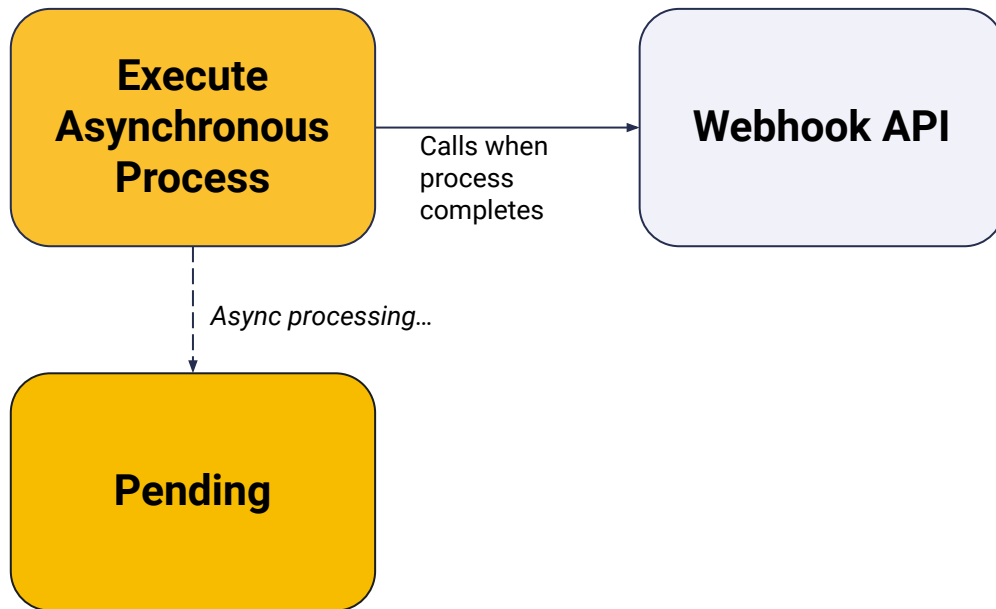
- Definition
- Flows

Webhooks are...

- often used in **asynchronous processing** to sync up different processes
- a lightweight API that helps **programs communicate**
- a **one way API** which will return status, but not a large payload
- called when the application **completes a process** or a **certain state is achieved** vs. a means of pulling data into an application
- a foundation for **automation**

What are Webhooks?

- Definition
- Flows





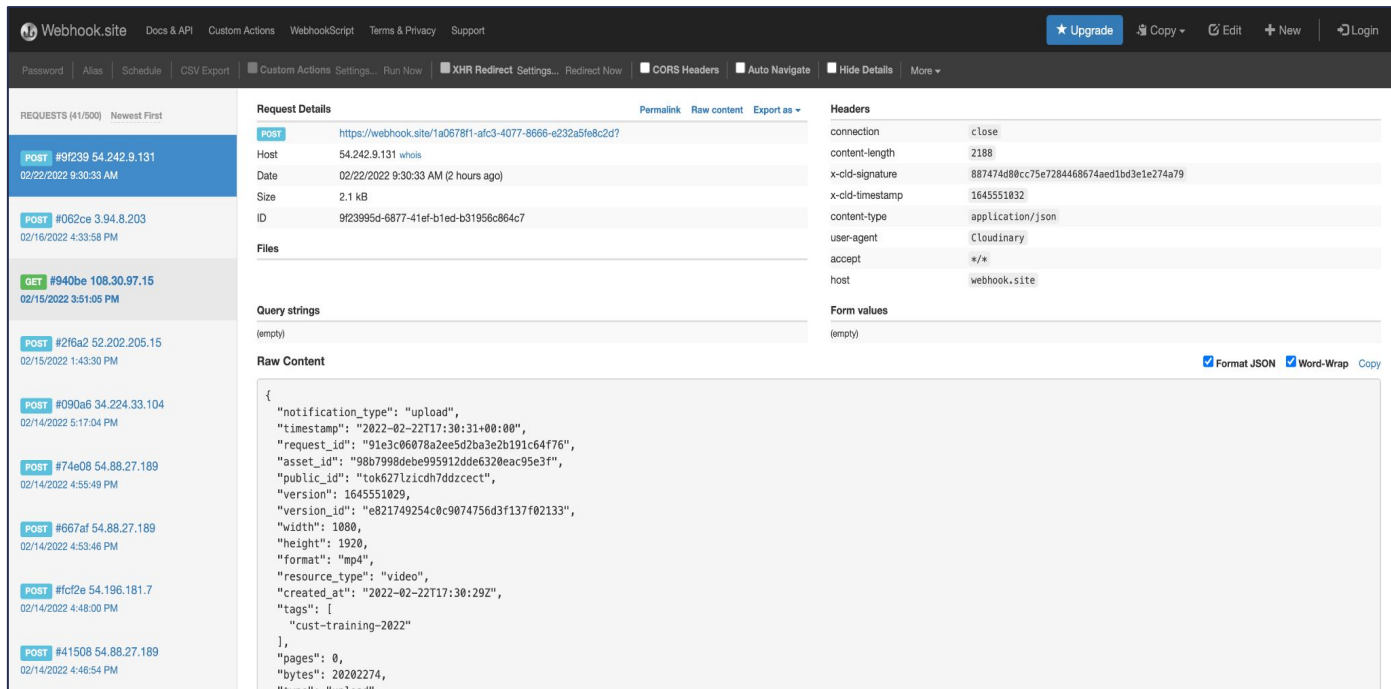
Upload API Examples

- Sprites
- Explode
- Multi
- Create Slideshow

Upload API Examples

- **webhook.site**
- DAM Global
- Notification URL
- Demo Response
- Tracking
- Sprite
- Explode and Sprite
- Multi
- Create video

<https://webhook.site>



The screenshot displays the Webhook.site interface. At the top, there's a navigation bar with links for Docs & API, Custom Actions, WebhookScript, Terms & Privacy, and Support. A blue 'Upgrade' button is visible on the right. Below the navigation bar, a toolbar contains various icons for actions like Copy, Edit, New, and Login. The main content area is divided into two panels. The left panel, titled 'REQUESTS (11/500) Newest First', lists several incoming requests with their methods (POST, GET), IDs, and timestamps. The right panel, titled 'Request Details', shows the details for a specific POST request. It includes the request URL, host, date, size, and ID. Below this, the 'Headers' section lists various headers like connection, content-length, x-cld-signature, x-cld-timestamp, content-type, user-agent, accept, and host. The 'Form values' section is currently empty. The 'Raw Content' section shows the raw JSON body of the request, which contains metadata about the upload, including notification type, timestamp, request ID, asset ID, public ID, version, and resource type (video).

Webhook.site Docs & API Custom Actions WebhookScript Terms & Privacy Support

Upgrade Copy Edit New Login

Password Alias Schedule CSV Export Custom Actions Settings... Run Now XHR Redirect Settings... Redirect Now CORS Headers Auto Navigate Hide Details More

REQUESTS (11/500) Newest First

POST #91239 54.242.9.131 02/22/2022 9:30:33 AM

POST #062ce 3.94.8.203 02/16/2022 4:33:58 PM

GET #940be 108.30.97.15 02/15/2022 3:51:05 PM

POST #216a2 52.202.205.15 02/15/2022 1:43:30 PM

POST #090a6 34.224.33.104 02/14/2022 5:17:04 PM

POST #74e08 54.88.27.189 02/14/2022 4:55:49 PM

POST #667af 54.88.27.189 02/14/2022 4:53:46 PM

POST #fcf2e 54.196.181.7 02/14/2022 4:48:00 PM

POST #41508 54.88.27.189 02/14/2022 4:46:54 PM

Request Details Permalink Raw content Export as

POST https://webhook.site/1a0678f1-afc3-4077-8666-e232a5fe8c2d7

Host 54.242.9.131 whois

Date 02/22/2022 9:30:33 AM (2 hours ago)

Size 2.1 kB

ID 9123995d-6877-41ef-b1ed-b31958c864c7

Files

Query strings

(empty)

Raw Content

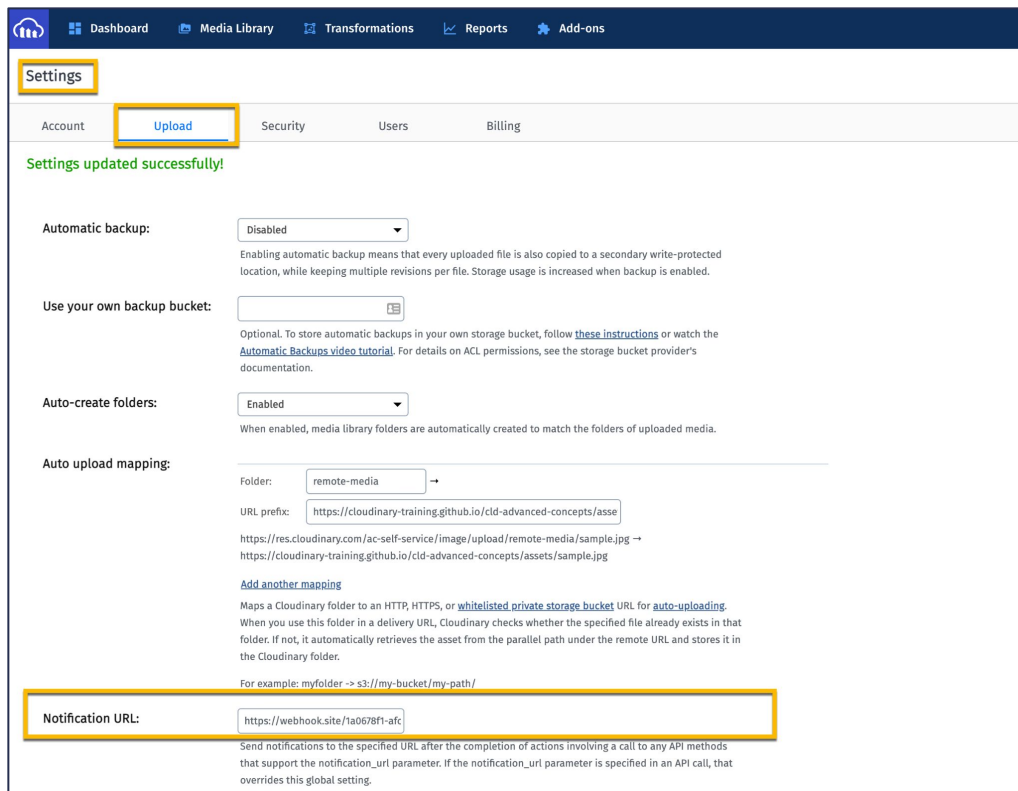
Format JSON Word-Wrap Copy

```
{
  "notification_type": "upload",
  "timestamp": "2022-02-22T17:30:31+00:00",
  "request_id": "91e3c06078a2ee5d2ba3e2b191c64f76",
  "asset_id": "98b7998debe995912d0e6320eac95e3f",
  "public_id": "tok627Lzicdh7ddzcect",
  "version": 1645551029,
  "version_id": "e821749254c0c9074756d3f137f02133",
  "width": 1080,
  "height": 1920,
  "format": "mp4",
  "resource_type": "video",
  "created_at": "2022-02-22T17:30:29Z",
  "tags": [
    "cust-training-2022"
  ],
  "pages": 0,
  "bytes": 20202274,
  "thumbnail_url": "https://res.cloudinary.com/cloudinary/image/upload/v1645551029/tok627Lzicdh7ddzcect/e821749254c0c9074756d3f137f02133.jpg"
}
```

Upload API Examples

- webhook.site
- DAM Global
- Notification URL
- Demo Response
- Tracking
- Sprite
- Explode and Sprite
- Multi
- Create video

Add a Notification URL



The screenshot shows the Cloudinary Settings page with the 'Upload' tab selected. The 'Settings' tab is also highlighted. A green message at the top says 'Settings updated successfully!'. The 'Automatic backup' is set to 'Disabled'. The 'Use your own backup bucket' option is disabled. The 'Auto-create folders' is set to 'Enabled'. The 'Auto upload mapping' section shows a folder named 'remote-media' mapped to a URL prefix. The 'Notification URL' field is highlighted with a yellow box and contains the value 'https://webhook.site/1a0678f1-afc'. Below this field, there is a description: 'Send notifications to the specified URL after the completion of actions involving a call to any API methods that support the notification_url parameter. If the notification_url parameter is specified in an API call, that overrides this global setting.'

Settings

Account Upload Security Users Billing

Settings updated successfully!

Automatic backup: Disabled

Enabling automatic backup means that every uploaded file is also copied to a secondary write-protected location, while keeping multiple revisions per file. Storage usage is increased when backup is enabled.

Use your own backup bucket: ☐

Optional. To store automatic backups in your own storage bucket, follow [these instructions](#) or watch the [Automatic Backups video tutorial](#). For details on ACL permissions, see the storage bucket provider's documentation.

Auto-create folders: Enabled

When enabled, media library folders are automatically created to match the folders of uploaded media.

Auto upload mapping:

Folder: remote-media →

URL prefix: https://cloudinary-training.github.io/cld-advanced-concepts/assets

https://res.cloudinary.com/ac-self-service/image/upload/remote-media/sample.jpg →
https://cloudinary-training.github.io/cld-advanced-concepts/assets/sample.jpg

[Add another mapping](#)

Maps a Cloudinary folder to an HTTP, HTTPS, or [whitelisted private storage bucket](#) URL for [auto-uploading](#). When you use this folder in a delivery URL, Cloudinary checks whether the specified file already exists in that folder. If not, it automatically retrieves the asset from the parallel path under the remote URL and stores it in the Cloudinary folder.

For example: myfolder -> s3://my-bucket/my-path/

Notification URL: https://webhook.site/1a0678f1-afc

Send notifications to the specified URL after the completion of actions involving a call to any API methods that support the notification_url parameter. If the notification_url parameter is specified in an API call, that overrides this global setting.

Upload API Examples

- webhook.site
- DAM Global Notification URL
- **Demo Response Tracking**
- Sprite
- Explode and Sprite
- Multi
- Create video

Test by Deleting an Asset in the DAM Media Library

Delete Response as it appears in
<https://webhook.site>

Raw Content

☒ Format JSON ☒ Word-Wrap [Copy](#)

```
{
  "notification_type": "delete",
  "resources": [
    {
      "resource_type": "image",
      "type": "upload",
      "asset_id": "f93dc448fe69c881f821aae29b707c95",
      "public_id": "koi",
      "version": 1644952831
    }
  ]
}
```

Upload API Examples

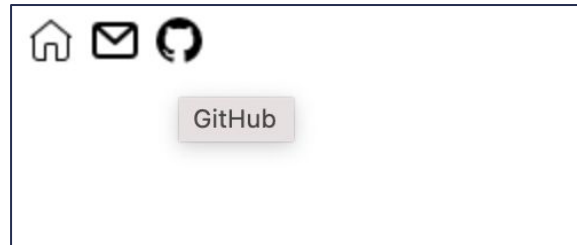
- webhook.site
- DAM Global Notification URL
- Demo Response Tracking
- **Sprite**
- Explode and Sprite
- Multi
- Create video

generate_sprite

- upload and tag a set of svg icons
- execute generate_sprite from icons
- use the CSS in the response to access the icons in HTML

CSS and Sprite Image

```
.webhooks-email, .webhooks-github, .webhooks-home {  
  background:  
    url('//res.cloudinary.com/cloudinary-training/image/sprite/v16  
45736404/iconduck.png') no-repeat;  
}  
.webhooks-email {  
  background-position: 0px 0px;  
  width: 24px;  
  height: 24px;  
}.webhooks-github {  
  background-position: 0px -26px;  
  width: 24px;  
  height: 24px;  
}.webhooks-home {  
  background-position: 0px -52px;  
  width: 24px;  
  height: 24px;  
}
```



Web Page with Sprite Icons
Sprite Image
Sprite CSS

Upload API Examples

- webhook.site
- DAM Global
Notification URL
- Demo Response
Tracking
- Sprite
- **Explode and
Sprite**
- Multi
- Create video

explode a GIF and generate_sprite

- upload a GIF
- use the explode action to access layers within the GIF
- create a sprite to pull a set of those layers into a PNG

GIF



Sprite URL

SPRITE



Upload API Examples

- webhook.site
- DAM Global
Notification URL
- Demo Response
Tracking
- Sprite
- Explode and
Sprite
- **Multi**
- Create video

multi

- upload and tag a set of images
- execute multi with a delay
- generate a GIF or mp4
- images are in alphabetical order of public id

Images



GIF



Upload API Examples

- [webhook.site](#)
- [DAM Global Notification URL](#)
- [Demo Response Tracking](#)
- [Sprite](#)
- [Explode and Sprite](#)
- [Multi](#)
- **Create video**

create_slideshow

- create a manifest with transitions, slideshow duration and duration for each asset
- assets can be image or video
- generate an mp4
- control order in manifest

Video



[Video Link](#)



Common Workflow Use Cases

- PIM Update After Eager Transformations Complete
- Upload Widget with Notification URL in Preset
- Moderation

Common Workflow Use Cases

- Update PIM Intro
- Update PIM Flow
- Upload Widget Intro
- Upload Widget Flow
- Moderation Flow

PIM Update Occurs in Webhook When Eager Transformation is Complete

- Add a Cloudinary image URL to you PIM after uploading
- A watermarked derived image will be served
- Use a webhook to update your PIM database with the image URL

POST Response to Webhook

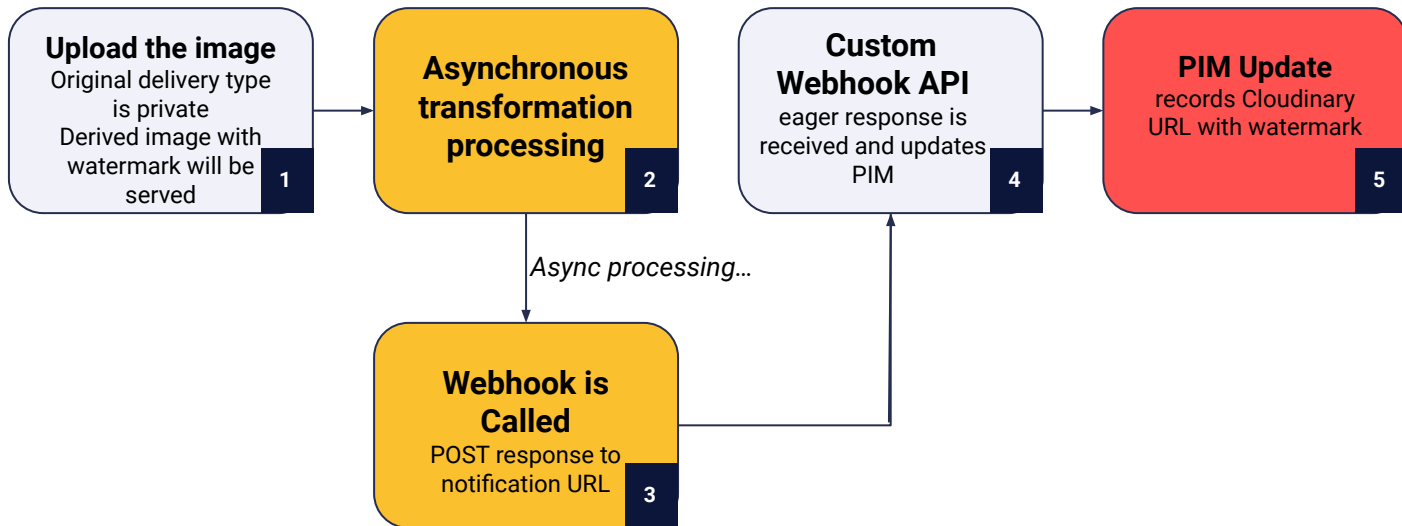
```
{
  "notification_type": "eager",
  "timestamp": "2022-02-24T21:56:29+00:00",
  "request_id": "f0227d2be25cd5a1f14da59c527cad5c",
  "eager": [
    {
      "transformation": "g_south_east,l_webhooks:logo,o_200,w_200,x_30,y_30",
      "width": 1920,
      "height": 2880,
      "bytes": 908315,
      "format": "jpg",
      "url": "http://res.cloudinary.com/cloudinary-training/image/private/g_south_east,l_webhooks:logo,o_200,w_200,x_30,y_30/v1645739702/webhooks/vase.jpg",
      "secure_url": "https://res.cloudinary.com/cloudinary-training/image/private/g_south_east,l_webhooks:logo,o_200,w_200,x_30,y_30/v1645739702/webhooks/vase.jpg"
    }
  ],
  "batch_id": "7c96871e44758be0ed1fe656d1efcbaae2e73dc5cfc3d790d3b66a36e0c48409",
  "asset_id": "b320b5bff40a1f4b7f7848a4bf1a5711",
  "public_id": "webhooks/vase"
}
```



Common Workflow Use Cases

- Update PIM Intro
- **Update PIM Flow**
- Upload Widget Intro
- Upload Widget Flow
- Moderation Flow

PIM Update Flow



```
{ "notification_type": "eager",  
  ... "eager": [  
  
    ...  
    "secure_url":  
    "https://res.cloudinary.com/cloudinary-training/image/private/g_south_east,l_webhooks:logo,o_100,w_400,x_30,y_30/v1645739702/w  
    ebhooks/vase.jpg"  
  ] }, ... }
```


Common Workflow Use Cases

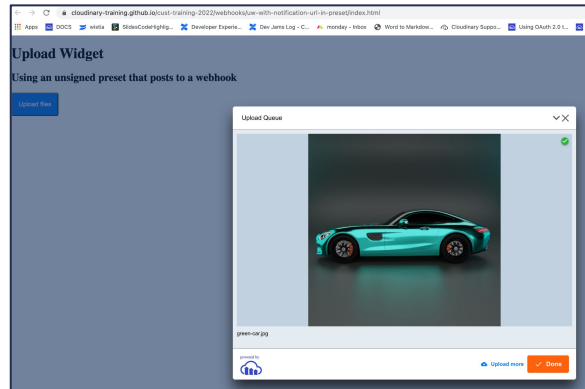
- Update PIM Intro
- Update PIM Flow
- **Upload Widget**
Intro
- Upload Widget
Flow
- Moderation Flow

Unsigned Preset with Notification URL in Upload Widget

- create a unsigned preset containing a notification_url named **'track-with-webhook'**
- use in front end upload with Upload Widget

```
cloudinary.api
.create_upload_preset({
  name: 'track-with-webhook',
  unsigned: true,
  notification_url: 'https://webhook.site/1a0678f1-afc3-4077-8666-e232a5fe8c2d'
})
```

code

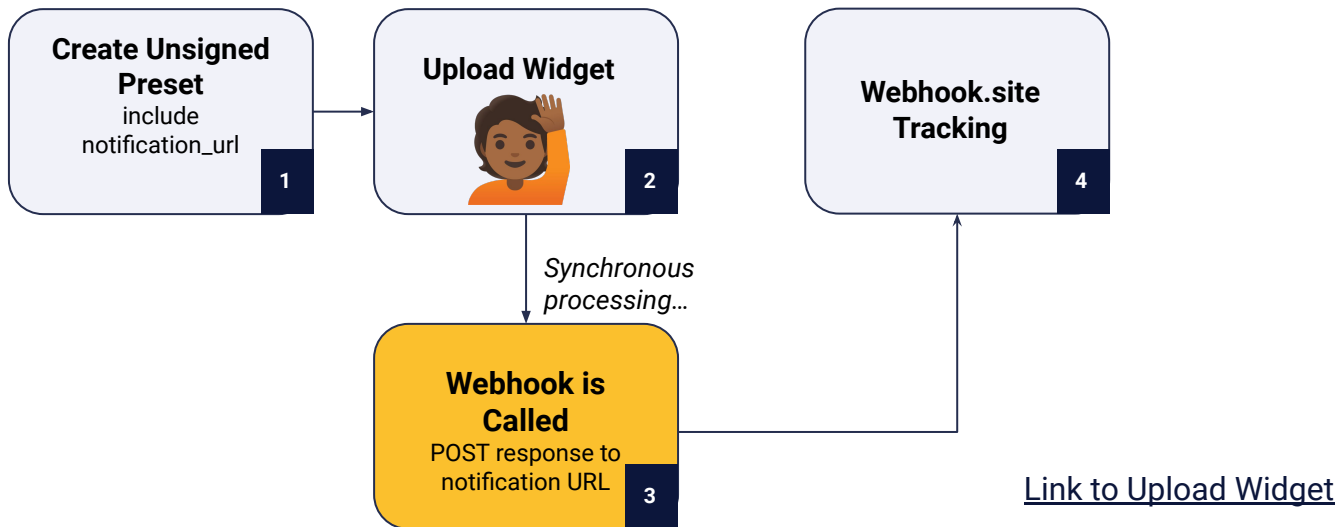


Web page with Upload Widget

Common Workflow Use Cases

- Update PIM Intro
- Update PIM Flow
- Upload Widget Intro
- **Upload Widget Flow**
- Moderation Flow

Front End Upload Widget Notification

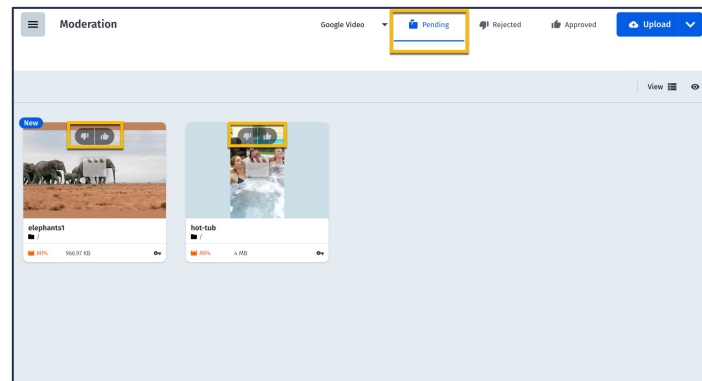
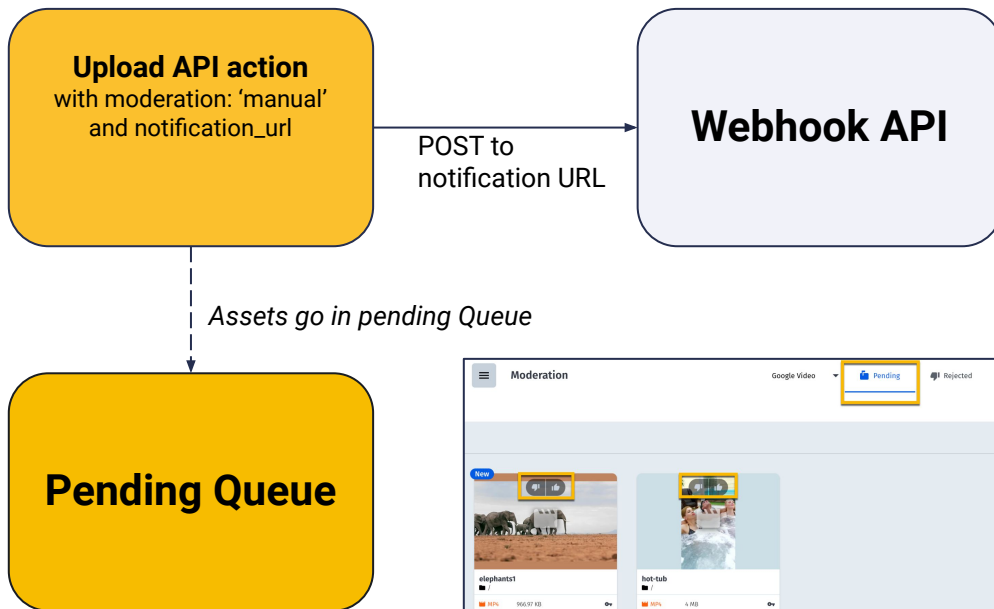


```
{
  "notification_type": "upload",
  "timestamp": "2022-03-04T19:51:23+00:00",
  ...
  "public_id": "xsahlpu1dte4xjwvcf5k",
  "version": 1646423483,
  "secure_url":
    "https://res.cloudinary.com/cloudinary-training/image/upload/v1646423483/xsahlpu1dte4xjwvcf5k.jpg"
  ...
}
```

Common Workflow Use Cases

- Update PIM Intro
- Update PIM Flow
- Upload Widget Intro
- Upload Widget Flow
- **Moderation Flow**

Moderation Flow





Custom Webhooks

- Email Notification for Eager Transformation
- Background Removal with Manual Moderation Walkthrough
- Managing Google AI Video Moderation Add-on Queues
- Face Recognition using Amazon Rekognition and Cloudinary Tagging

Custom Webhooks

- Email Notification
- Upload API Add-ons
- Background Removal Moderation
- Google AI Video Moderation
- Amazon Rekognition Tagging

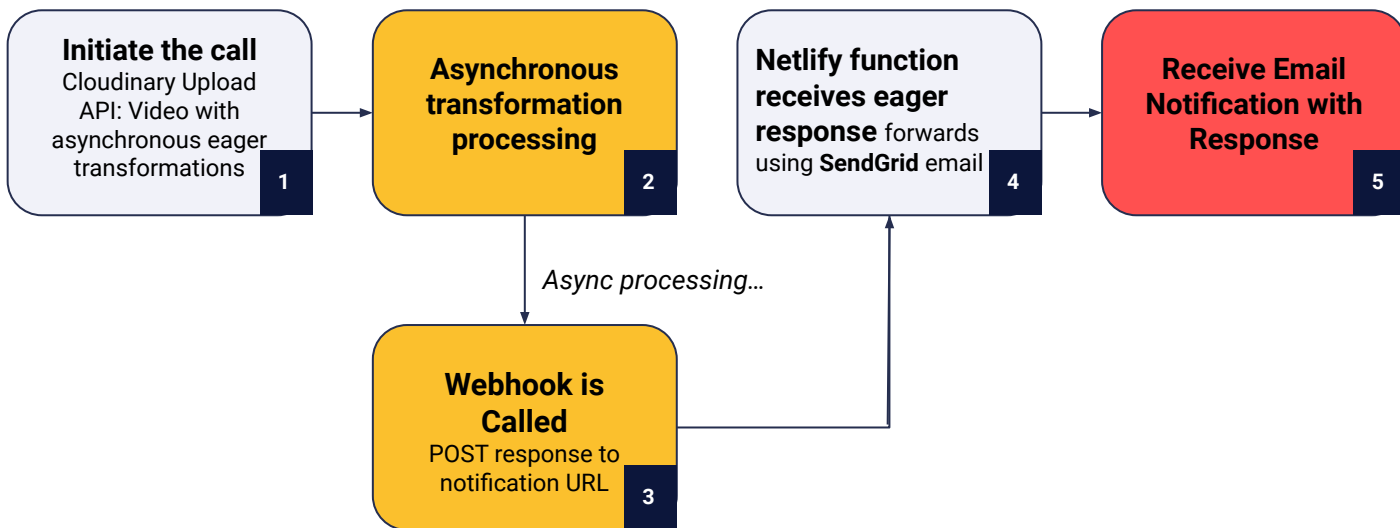
Process Video Eagerly and Email Response

- Create an API using a Netlify function
- Use a **Netlify** function as webhook
- **Netlify** function uses **SendGrid API** to send response values in Email
- Upload with **eager_async**
- Set **eager_notification_url** value to Netlify function URL
- **Eager response** will be posted to Netlify function
- SendGrid recipients will receive eager response in Email

Custom Webhooks

- Email Notification
- Upload API Add-ons
- Background Removal
- Google AI Video Moderation
- Amazon Rekognition Tagging

Look at Send Email Flow



Custom Webhooks

- Email Notification
- Upload API Add-ons
- Background Removal Moderation
- Google AI Video Moderation
- Amazon Rekognition Tagging

Webhook API Code

```
exports.handler = async function (event, context) {  
  const data = JSON.parse(event.body);  
  sgMail.setApiKey(process.env.SENDGRID_API_KEY);  
  
  const response = await sgMail.sendMultiple({  
    to: process.env.TO_RECIPIENTS.split(' '),  
    from: process.env.FROM_VERIFIED_SENDER,  
    subject: 'Webhook Notification',  
    text: JSON.stringify(data, null, 2),  
    trackingSettings: {  
      clickTracking: {  
        enable: false  
      }  
    }  
  });  
  return {  
    statusCode: response[0].statusCode,  
    body: JSON.stringify({ message: response[0] }),  
  };  
};
```

1. add **handler** to **exports** object that references an **async** function
2. `JSON.parse` to get POST data from `event.body`
3. register the SENDGRID API KEY
4. use **await** to call **sendMultiple** and pass the email message
5. return a **status code** and a string **message**

Custom Webhooks

- Email Notification
- **Upload API Add-ons**
- Background Removal Moderation
- Google AI Video Moderation
- Amazon Rekognition Tagging

Upload API Add-ons

- Cloudinary AI Background Removal
- Google AI Video Moderation
- Amazon Rekognition
- Web Purify Image Moderation
- Meta Defender Anti-Malware Protection

Custom Webhooks

- Email Notification
- Upload API Add-ons
- **Background Removal Moderation**
- Google AI Video Moderation
- Amazon Rekognition Tagging

Background Removal with Manual Moderation

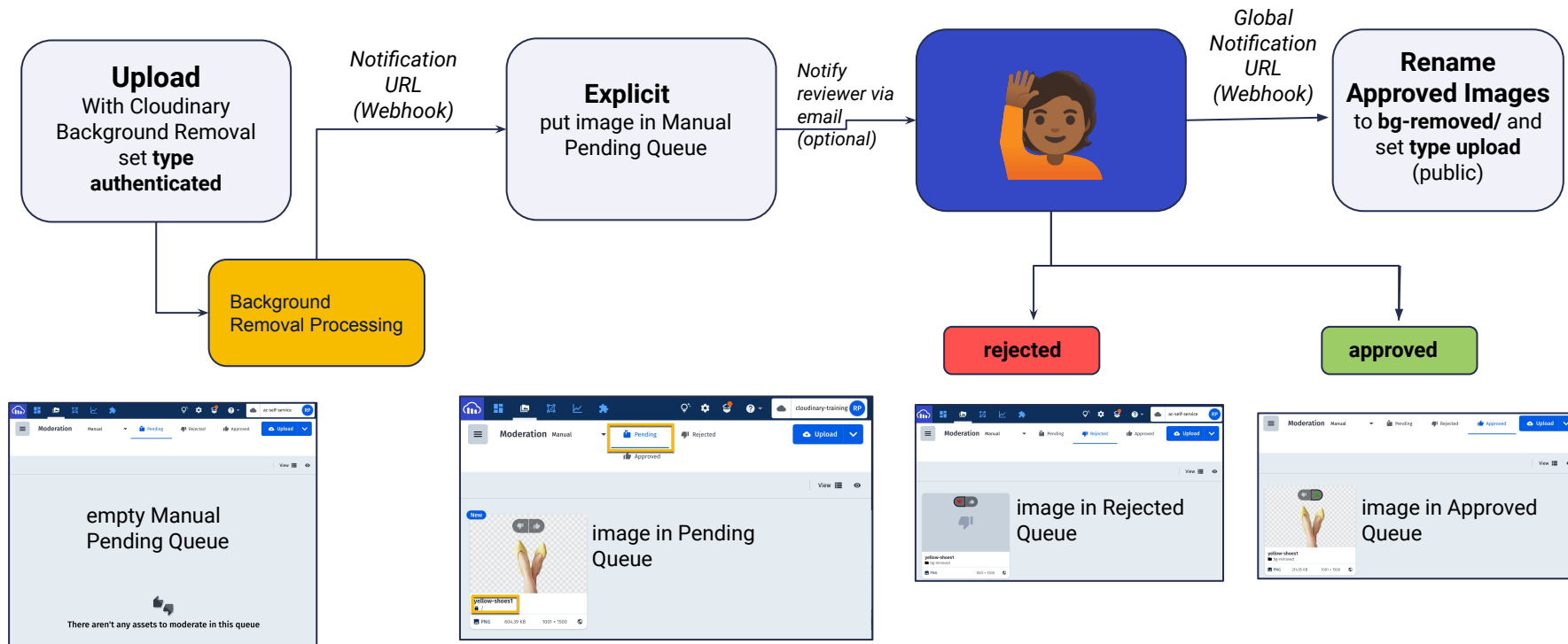
Process with Human Interaction

1. Enable the **Cloudinary AI Background Removal Add-on**
2. Add Webhook to **Global Notification** URL in DAM
3. **Execute Upload Image using Background Removal** with type Authenticated and call Webhook to explicitly **put the image in the Manual Moderation pending queue** and specify a webhook
4. At this point you could email the reviewer to approve or reject
5. **Human approves** the image and image moves to approval queue
6. When the image is approved, **Media Library triggers global webhook** defined in Settings
7. Global webhook gets images from approval queue and **moves them to 'bg-removed' folder** with 'upload' type so that they are **publicly available**



Cloudinary AI Background
Removal

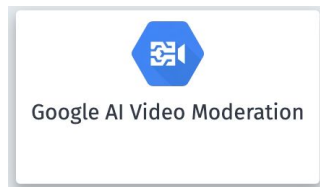
Background Removal With Manual Moderation



Custom Webhooks

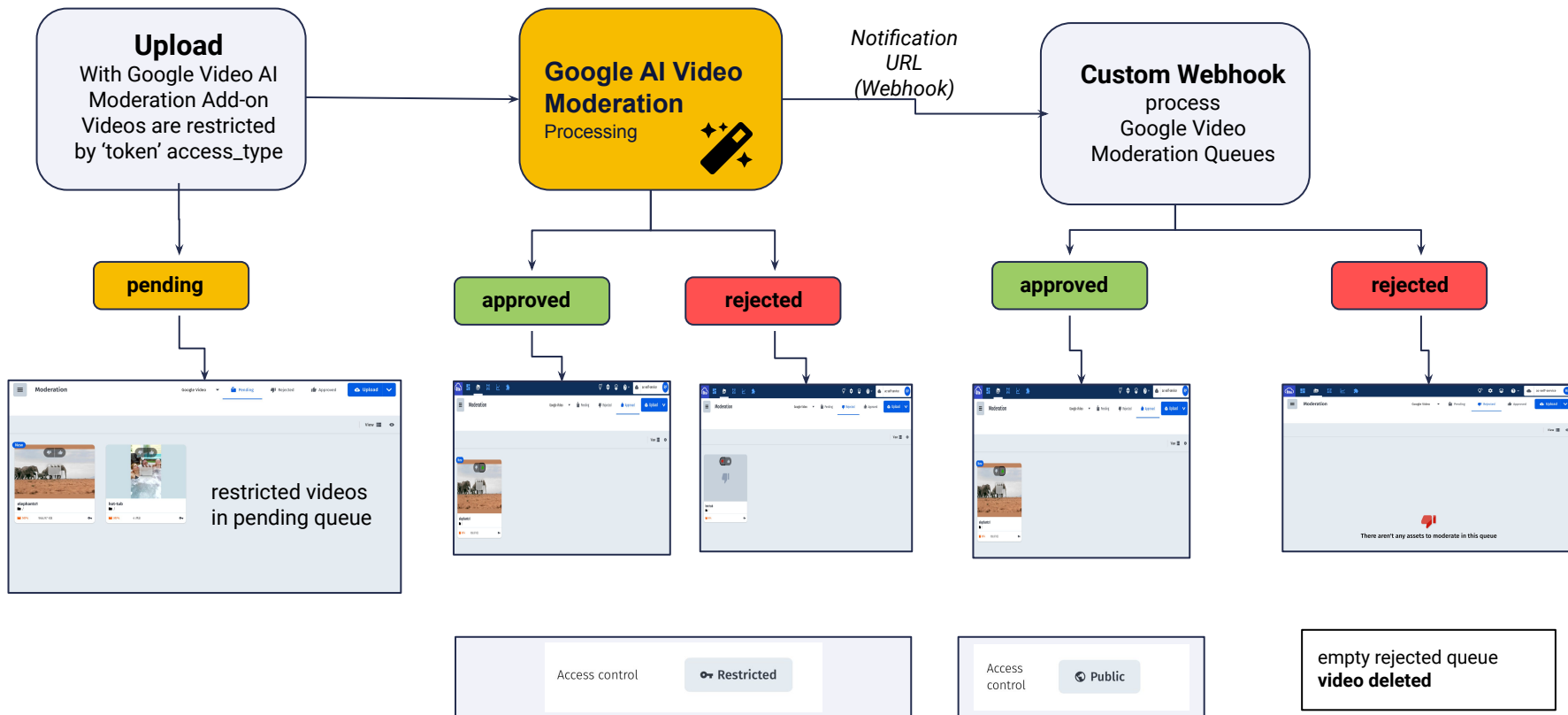
- Email Notification
- Upload API Add-ons
- Background Removal Moderation
- **Google AI Video Moderation**
- Amazon Rekognition Tagging

Upload Video with Google AI Video Moderation



- Goal is to share user videos on our website so we provide video upload and used Google AI for moderation
- Upload API with Google Video AI Moderation add-on
- upload using **access_control with access_type `token`** to prevent the URL from being accessed before moderation is complete
- **Custom webhook** called when moderation complete
- The webhook checks the Google moderation queues and **deletes the rejected videos** and **updates the approved videos access_type to 'anonymous'** so that the video is no longer restricted

Google AI Video Moderation Flow

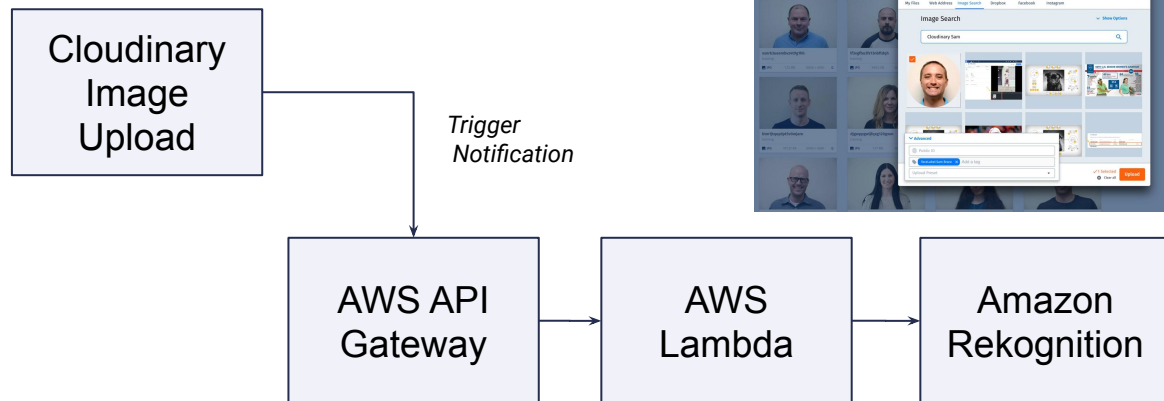


Custom Webhooks

- Email Notification
- Upload API Add-ons
- Background Removal
- Moderation
- Google AI Video Moderation
- **Amazon Rekognition Tagging**

Face Recognition using Amazon Rekognition and Cloudinary Tagging

- Use Cloudinary Tags, Upload API notification URLs, and AWS Lambdas to index and search for images
- Use Case: Facial recognition of employees





Webhook Security

- Verifying Notification Signatures
- API Header Verification

Webhook Security

- API Header Verification
- Verifying Cloudinary Notification Signatures

API Header Verification

Look at Header **httpMethod**, **referrer**, **origin** and **host** and fail if not what is expected

```
if (!event.body || event.httpMethod !== 'POST') {  
  return {  
    statusCode: 400,  
    headers: {  
      'Access-Control-Allow-Origin': '*',  
      'Access-Control-Allow-Headers':  
        'Content-Type',  
    },  
    body: JSON.stringify({  
      status: 'invalid-method',  
    }),  
  };  
}
```

```
if (event.headers.referrer.includes('mydomain.com'))  
{  
  // process the function  
} else {  
  return {  
    statusCode: 401,  
    body: JSON.stringify('Unauthorized')  
  }  
}
```

But Cloudinary offers more security information in the response headers =>

Webhook Security

- API Header Verification
- Verifying Cloudinary Notification Signatures

Verifying Cloudinary Notification Signatures

- Cloudinary signs all notification url responses with
 - x-cld-signature
 - x-cld-timestamp
 - content-type
- validate with utility function and check that signature was generated within last 2 hours

| Headers | |
|-----------------|---|
| connection | close |
| content-length | 744 |
| x-cld-signature | 10d757b9d89ffe455d460dddfc1c56c14464c68 |
| x-cld-timestamp | 1645748945 |
| content-type | application/json |
| user-agent | Cloudinary |
| accept | */* |
| host | webhook.site |

```
cloudinary.utils.verifyNotificationSignature(body, timestamp, signature, valid_for)
```


Thank You.

Resources

- [Cloudinary Training on GitHub](#)
 - [Customer Ed 2022 Webhooks](#)
 - [Cloudinary Custom Webhooks Deployed to Netlify](#)
- [Verifying Notification Signatures](#)
- [Securing Netlify Functions](#)
- [Cloudinary Media Flows](#)
- [How to Use Amazon Rekognition on Cloudinary to Auto-Tag Faces with Names](#)
- [Eager Transformations](#)
- [Webhooks, upload notifications and background image processing](#)
- [Background Removal](#)
- [Google AI Video Moderation](#)
- Other Low Code Platforms: [AWS Chalice](#), [Serverless](#), [Vercel Functions](#)