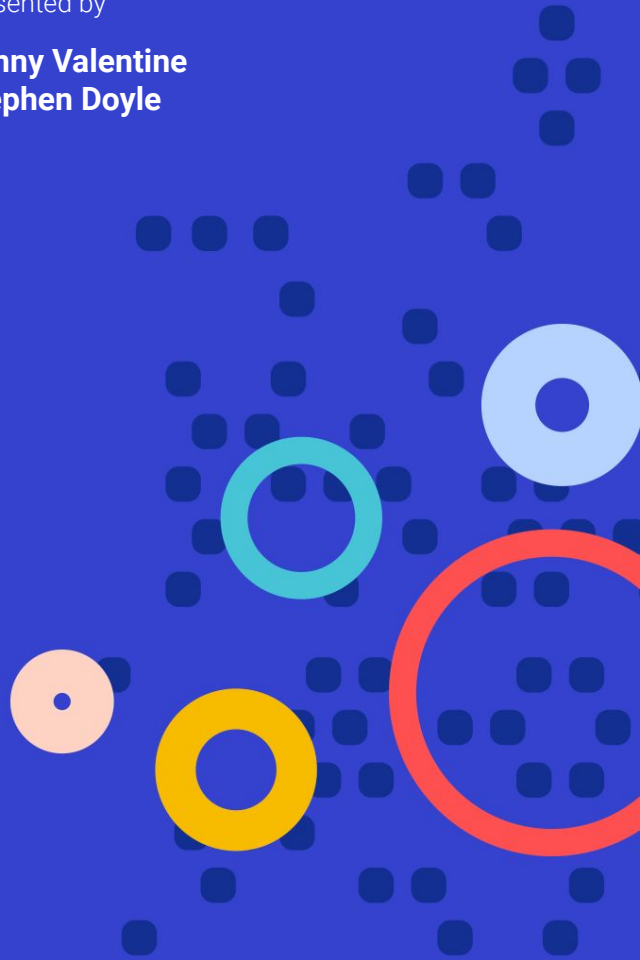
Cloudinary

Presented by

Danny Valentine
Stephen Doyle

Introduction to Cloudinary for API Users and Node.js Developers

Overview of Cloudinary's Node.js SDK



Topics

- Environment Setup/ Account Identifiers
- Uploading Basics
- Managing Uploaded Assets
- Transforming Assets
- Creating Presets and Named Transformations
- Setting Up Backups
- Fine Tuning Your Account Settings
- Next Steps for Further Support

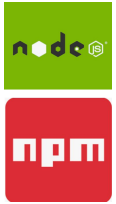
Environment Setup

- Install Node.js
- Install IDE
- Identify Credentials
  - **Cloud Name**
  - **API Key & API Secret**
- Set Up Credentials
- Running Code

Environment Setup

- **Install Node.js**
- Install IDE
- Identify Credentials
- Set Up Credentials
- Run Code

Install Node.js and NPM



You will need to install Node.js on your machine, version 10 or higher. Installing Node.js will also install npm, the package manager for Node.js

Mac Users

```
$ brew install node
```

We recommend using HomeBrew:
https://nodejs.org/tr/download/package-manager/#macos

Windows Users

https://nodejs.org/en/download/

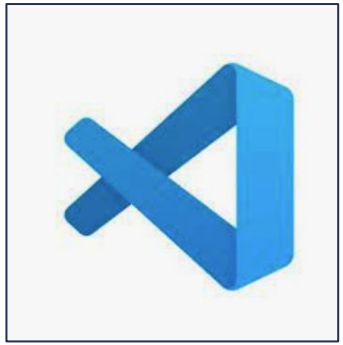
```
# verify versions
$ node --version
v16.13.0

$ npm --version
8.1.0
```

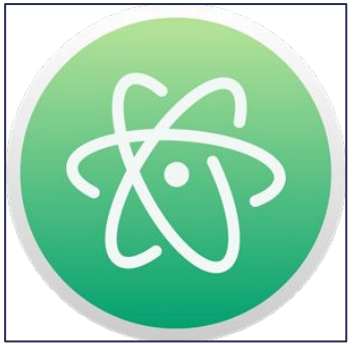
Environment Setup

- Install Node.js
- **Install IDE**
- Identify Credentials
- Set Up Credentials
- Run Code

Install IDE

Many options, including text editor.



GitHub Repository

https://github.com/cloudinary-training/cld-intro-nodejs

Environment Setup

- Install Node.js
- Install IDE
- **Identify Credentials**
- Set Up Credentials
- Run Code

Identify Credentials - API Key is public and API Secret is private

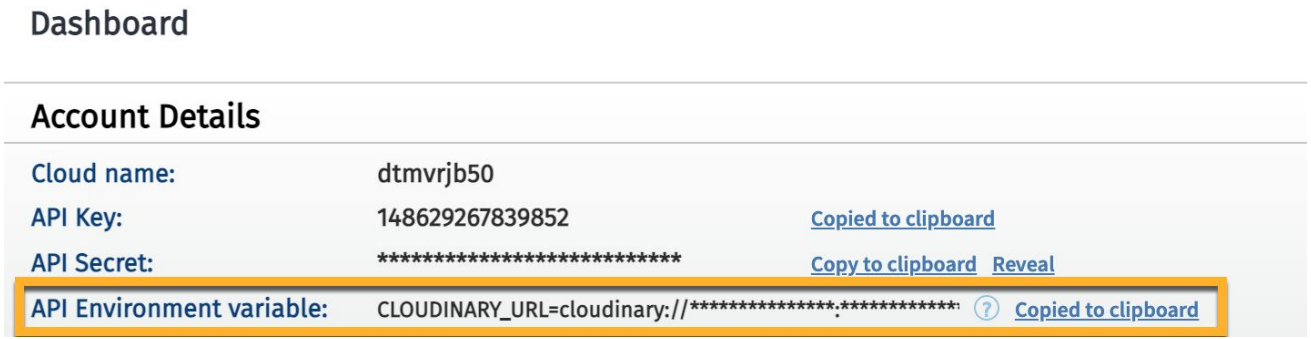
Auto-generated access identifiers.

Important!
Do not expose your API Secret in your client-side code.

Used for communicating with Cloudinary's API and signing your requests.

We'll use the API Environment Variable

Key: **CLOUDINARY_URL** **Value**: **cloudinary://API_KEY:API_SECRET@CLOUD_NAME**



Environment Setup

- Install Node.js
- Install IDE
- Identify Credentials
- **Set Up Credentials**
- Run Code

Cloudinary

Set Up Credentials

For this training we'll store the CLOUDINARY_URL in a **.env** file in the root of the project. You can copy `.env.sample` and save it as `.env` with your own Cloudinary URL

```
CLOUDINARY_URL=cloudinary://my_key:my_secret@my_cloud_name
```

We'll use the npm.js dotenv library to read the data out of your **.env** file into process.env

```
npm i dotenv
```

Environment Setup

- Install Node.js
- Install IDE
- Identify Credentials
- **Set Up Credentials**
- Run Code

Cloudinary

Set Up Credentials

We can test everything is set up by creating a test script called `testCredentials.js`.

Within this script, we require the dotenv package. It's advised to include this line as early as possible in your application

```
require('dotenv').config();
```

We'll then install the npm cloudinary library with `npm i cloudinary`, and include it in our script. When we execute config(), the API_SECRET, API_KEY, and CLOUD_NAME will each be accessible in the script.

```
const cloudinary = require('cloudinary').v2;
console.log(cloudinary.config().cloud_name);
console.log(cloudinary.config().api_key);
console.log(cloudinary.config().api_secret);
```

Environment Setup

- Install Node.js
- Install IDE
- Identify Credentials
- Set Up Credentials
- **Run Code**

Run Code

```
node testCredentials.js
```

Output

```
my_cloud_name
my_key
my_secret
```

Uploading Basics

- Basic Upload Method
  - **Response**
  - **Alternate Path**
- Resource Type
- Public ID
  - **Naming Options**
- Folder

Uploading Basics

- **Basic Upload Method**
  - Response
  - Alternate Path
- Resource Type
- Public ID
  - Naming Options
- Folder
- Upload Source Options

Basic Upload Method

Performs an authenticated upload API call over HTTPS while sending the file.

The upload returns a promise.

```
cloudinary.uploader.upload("./assets/cat.jpg")
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Included in the URL. 

Examples found in `upload.js`

Uploading Basics

- **Basic Upload Method**
  - **Response**
  - Alternate Path
- Resource Type
- Public ID
  - Naming Options
- Folder
- Upload Source Options

Cloudinary

Response

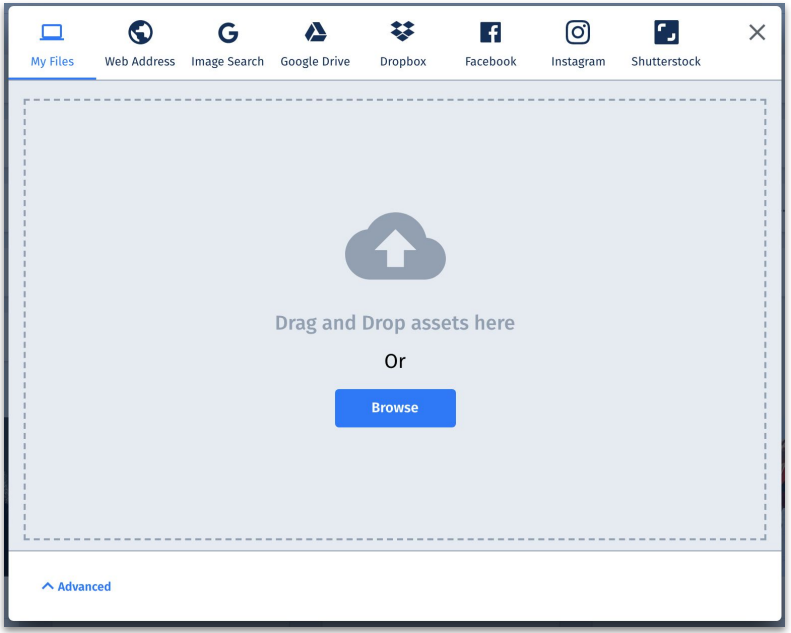
```
{
  asset_id: '486f17d377628cdc44f8160a97afa2fe',
  public_id: 'cat',
  version: 1639061268,
  version_id: '50a37d2891bf23530f093e116e071acc',
  signature: '13a51a842d34dd1807bf5f9fce46c95ef1bf57de',
  width: 2048,
  height: 1536,
  format: 'jpg',
  resource_type: 'image',
  created_at: '2021-12-09T14:47:48Z',
  tags: [],
  pages: 1,
  bytes: 379984,
  type: 'upload',
  etag: 'fbb0f017dba3395f026e91f23f9fbd2f',
  placeholder: false,
  url: 'http://res.cloudinary.com/dannyv-training/image/upload/v1639061268/cat.jpg',
  secure_url: 'https://res.cloudinary.com/dannyv-training/image/upload/v1639061268/cat.jpg',
  access_mode: 'public',
  original_filename: 'cat',
  api_key: '383521115725668'
} undefined
```

Uploading Basics

- **Basic Upload Method**
  - Response
  - **Alternate Path**
- Resource Type
- Public ID
  - Naming Options
- Folder
- Upload Source Options

Basic Upload Method - Alternate Path

Can also be done via our DAM, an interactive web interface to add images and videos to your media library.



Uploading Basics

- Basic Upload Method
  - Response
  - Alternate Path
- **Resource Type**
- Public ID
  - Naming Options
- Folder
- Upload Source Options

Cloudinary

Resource Type

Indicates whether you want to upload an image, video or raw file.

Let Cloudinary determine the type by setting the parameter to “auto” or manually define for each file.

```
cloudinary.uploader.upload("./assets/dog.webm", {resource_type: "auto"})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

```
cloudinary.uploader.upload("./assets/dog.webm", {resource_type: "video"})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

```
cloudinary.uploader.upload("./assets/BLKCHCRY.TTF",
   {resource_type: "raw", use_filename: true, unique_filename: false})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Included in the URL. https://res.cloudinary.com/dem /image/ pload/sample.jpg

Uploading Basics

- Basic Upload Method
  - Response
  - Alternate Path
- Resource Type
- **Public ID**
  - Naming Options
- Folder
- Upload Source Options

Public ID

A unique identifier for your asset that appears in the URL.

Used to reference the uploaded resource, as well as for building dynamic delivery and transformation URLs.

```
res.cloudinary.com/sambrace/image/upload/v1583856029/kitten.jpg
```

If you don't supply a Public ID in the Upload API call, one will be randomly assigned in the response.

```
res.cloudinary.com/sambrace/image/upload/v1598402406/amvcr9bh5btj18jsgd4h.jpg
```

Uploading Basics

- Basic Upload Method
  - Response
  - Alternate Path
- Resource Type
- **Public ID**
  - **Naming Options**
- Folder
- Upload Source Options

Cloudinary

Public ID - Naming Options

There are several methods for naming your assets on upload.

You can use the asset's file name, but add random characters as a suffix.

This will help prevent overwriting a previously uploaded version.

```
cloudinary.uploader.upload("./assets/cat.jpg", {use_filename: true,
unique_filename: true})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

You can also retain the asset's file name as its Public ID.

```
cloudinary.uploader.upload("./assets/cat.jpg", {use_filename: true,
unique_filename: false})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

You can manually define a Public ID for the asset too.

```
cloudinary.uploader.upload("./assets/cat.jpg", {public_id: "ZeldaCat"})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Uploading Basics

- Basic Upload Method
  - Response
  - Alternate Path
- Resource Type
- Public ID
  - Naming Options
- **Folder**
- Upload Source Options

Folder

Cool Tip!
A folder will be automatically created for you with these calls, if it did not already exist.

A way to organize and divide your files.

Folder paths can be included in the `public_id` parameter or in a separate `folder` parameter.

```
cloudinary.uploader.upload("./assets/cheesecake.jpg", {public_id:
"food/my_favorite/cheesecake"})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

```
cloudinary.uploader.upload("./assets/dog.jpg", {folder: "pets/my_favorite"})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Included in the URL.

```
res.cloudinary.com/sambrace/image/upload/v1583856029/cute_animals/kitten.jpg
```

Uploading Basics

- Basic Upload Method
  - Response
  - Alternate Path
- Resource Type
- Public ID
  - Naming Options
- Folder
- **Upload Source Options**

Cloudinary

Upload Source Options

HTTP or HTTPS URL

```
cloudinary.uploader.upload("https://upload.wikimedia.org/wikipedia/commons/5/56/Chocolate_cupcakes.jpg")
    .then(result => {console.log(result)})
    .catch(error => {console.log(error)});
```

Private Storage URL (e.g. Amazon S3 or Google Cloud Storage)

```
cloudinary.uploader.upload("s3://my-bucket/my-path/example.jpg")
    .then(result => {console.log(result)})
    .catch(error => {console.log(error)});
```

Base64 Data URI

```
cloudinary.uploader.upload("")
    .then(result => {console.log(result)})
    .catch(error => {console.log(error)});
```

FTP URL

```
cloudinary.uploader.upload("ftp:user1:mypass@ftp.example.com/sample.jpg")
    .then(result => {console.log(result)})
    .catch(error => {console.log(error)});
```

Cloudinary Utilities

- Create a URL
- Create an Image Tag
- Create a Video Tag

Cloudinary Utilities

- **Create a URL**
- Create an Image Tag
- Create a Video Tag

Create a URL

Just add Cloudinary **public_id** as argument

```
cloudinary.url("food/my_favorite/cheesecake")
cloudinary.url("dog", {resource_type: "video"})
```

Result is a URL string:

```
http://res.cloudinary.com/cloud_name/image/upload/food/my_favorite/cheesecake
http://res.cloudinary.com/cloud_name/video/upload/dog
```

When you request, Cloudinary will cache and return the asset

Examples found in `utilities.js`

Cloudinary Utilities

- Create a URL
- **Create an Image Tag**
- Create a Video Tag

Create an Image Tag

Just add Cloudinary **public_id** as argument

```
cloudinary.image("food/my_favorite/cheesecake")
```

Result is an image tag string:

```
<img
src='http://res.cloudinary.com/dannyv-training/image/upload/v1/food/my_fav
orite/cheesecake' />
```

When you render from web page, Cloudinary will cache and return the asset in the image tag.

Cloudinary Utilities

- Create a URL
- Create an Image Tag
- **Create a Video Tag**

Create an Video Tag

Just add Cloudinary **public_id** as argument

```
cloudinary.video("dog")
```

Result is a video tag string:

```
<video poster='http://res.cloudinary.com/dannyv-training/video/upload/dog.jpg'>
  <source src='http://res.cloudinary.com/dannyv-training/video/upload/dog.webm' type='video/webm'>
  <source src='http://res.cloudinary.com/dannyv-training/video/upload/dog.mp4' type='video/mp4'>
  <source src='http://res.cloudinary.com/dannyv-training/video/upload/dog.ogv' type='video/ogg'>
</video>
```

When you render from web page, Cloudinary will cache and return the asset used by the browser in the video tag.

Remote Upload

- Auto-upload
- Fetch

Remote Upload

- **Upload Source Options**
  - **Fetch**
  - Auto Upload

Upload Source Options - Fetch

On-the-fly manipulation of existing remote images and optimized delivery via a CDN.

- Supported for images only.
- Cached on your Cloudinary account for performance reasons.
- Remote images are checked on a regular basis, and if the remote image changes, the cached image is updated accordingly.

```
cloudinary.url("https://upload.wikimedia.org/wikipedia/commons/8/84/Chocolate_Cake_Flourless_%281%29.jpg", {type: "fetch"})
```

Examples found in `remote.js`

Remote Upload

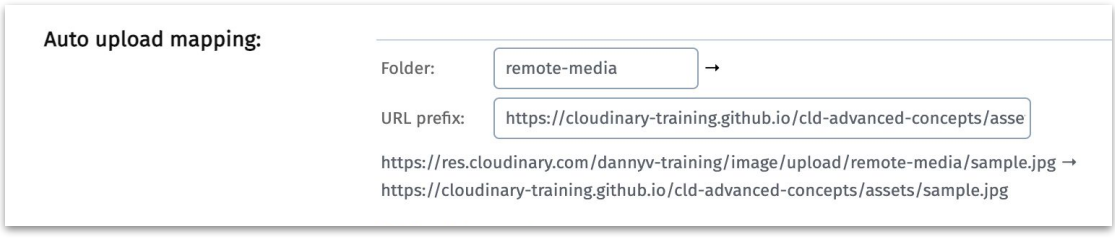
- **Upload Source Options**
  - Fetch
  - **Auto Upload**

Upload Source Options - Auto Upload

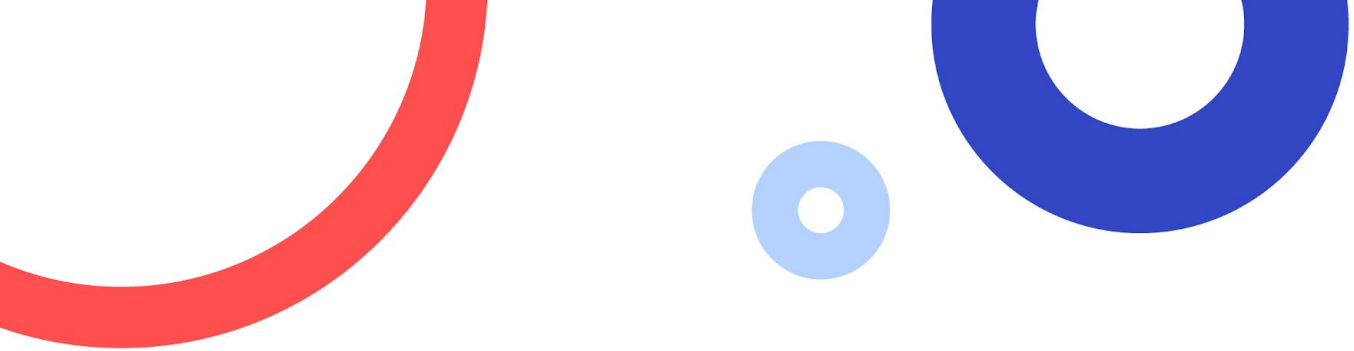
Combine the advantages of dynamic fetching from existing online locations, with the advantages of managing assets in your account.

Implemented by mapping a base remote URL to a specified folder in your Cloudinary account.

```
https://cloudinary-training.github.io/cld-advanced-concepts/assets/images/dolphin.jpg
```



```
cloudinary.url("remote-media/images/dolphin.jpg", {transformation: [{width: "300", zoom: "300", crop: "thumb"}]})
```



Managing Uploaded Content

- Browse
- Rename
- Delete
- Tag
- Invalidate

Examples found in `manage.js`

Managing Uploaded Content

- **Browse**
- Rename
- Delete
- Tag
- Invalidate

Browse

Using Admin API methods, you can list all of your uploaded assets with different criteria.

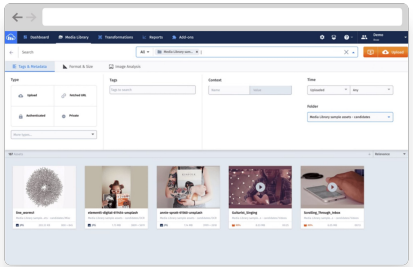
List all assets

```
cloudinary.api.resources()
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

List all images with a given prefix

```
cloudinary.api.resources({ type: 'upload',  prefix: 'sample' })
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

This is also available through the Advanced Search in the Console.



Cloudinary

Managing Uploaded Content

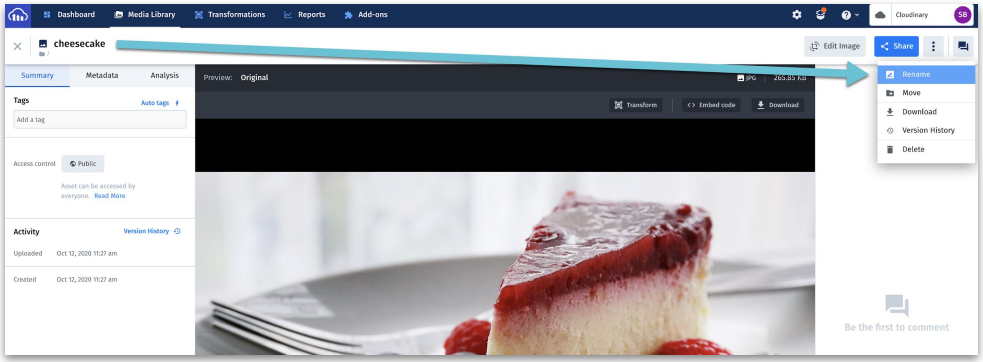
- Browse
- **Rename**
- Delete
- Tag
- Invalidate

Rename

Immediate and permanent updates in your cloud storage.

Any existing URLs of renamed assets and their associated derived assets are no longer valid.

```
cloudinary.uploader.rename("cheesecake", "my_cheesecake", {overwrite: true})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```



Cloudinary

Managing Uploaded Content

- Browse
- Rename
- **Delete**
- Tag
- Invalidate

Delete

Immediate and permanent updates in your cloud storage with the `destroy` method.

Any existing URLs of deleted assets and their associated derived assets are no longer valid.

```
cloudinary.uploader.destroy("my_cheesecake", {invalidate: true})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

You can also delete multiple assets at one time by their Public ID or even a prefix in the Public ID with our Admin API.

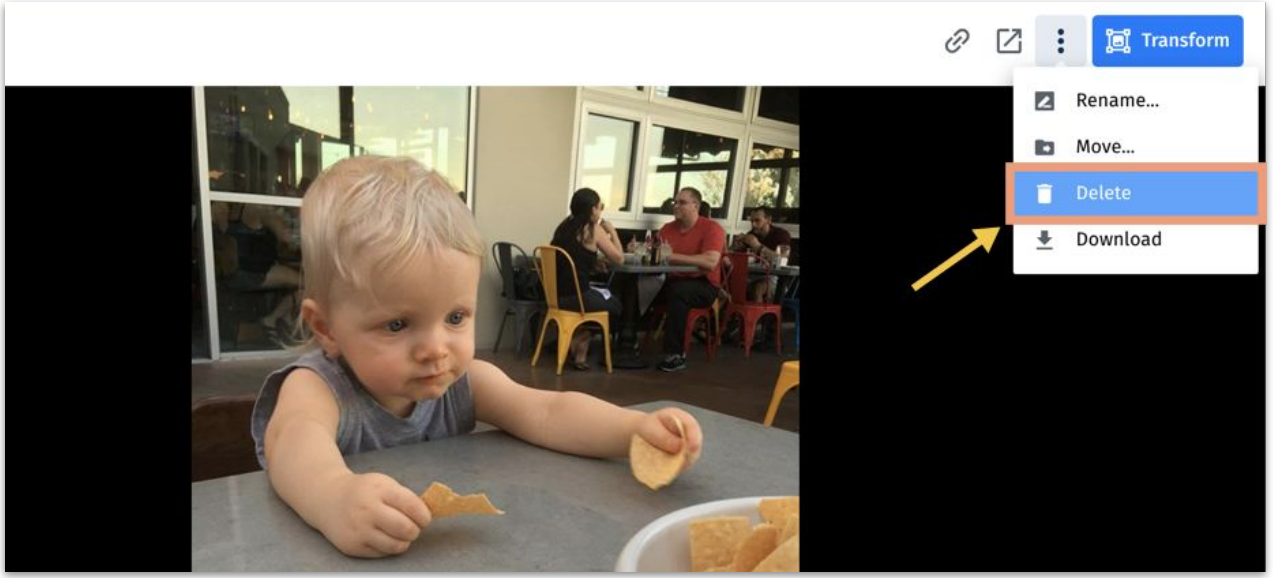
```
cloudinary.api.delete_resources(["squirrel", "sheep"])
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Managing Uploaded Content

- Browse
- Rename
- **Delete**
- Tag
- Invalidate

Delete

Deleting via the Upload API allows you to remove assets from your Cloud without having to manually delete them via the Media Library



Managing Uploaded Content

- Browse
- Rename
- Delete
- **Tag**
- Invalidate

Cloudinary

Tag

Categorize and organize your assets.

You can add tags to your assets at any time, whether during upload or after they are in your account.

```
cloudinary.uploader.upload("./assets/tag.jpg", {tags: "object, literally a tag"})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

```
cloudinary.uploader.add_tag("animal", [ "cat", "dog" ])
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

You can also remove tags from uploaded assets.

```
cloudinary.uploader.remove_all_tags([ "cat", "dog" ])
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Managing Uploaded Content

- Browse
- Rename
- Delete
- Tag
- **Invalidate**

Invalidate

Prevent users from accessing deleted or renamed assets by sending an invalidation request.

- This instructs the CDN to remove cached copies of the old asset.
- Not using this method may cause the old cached media asset can remain on the CDN servers for up to 30 days.

```
cloudinary.uploader.rename("cat", "cute_cat", {invalidate: true})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

The next request for the asset will pull the newest copy from your Cloudinary storage, or will return an error if the asset no longer exists.

This same approach can be done with versioning too, bypassing the CDN cached version and forcing delivery of the newest asset.

Transforming Content for Optimization

- Resizing
- Cropping
- Gravity
- Format
- Quality

Please run `node transforming_prerequisites.js` before trying the examples found in `transforming.js`

Transforming Content for Optimization

- **Resizing**
- Cropping
- Gravity
- Format
- Quality

Resizing

Change an asset's size by editing its width (w_) and/or height (h_)

Decreasing the width and/or height of the asset will commonly decrease the file size, optimizing it for a specific project.

```
cloudinary.url("cheesecake.jpg", {transformation: {width: 300, crop: "scale"});
http://res.cloudinary.com/dannyv-training/image/upload/c_scale,w_300/cheesecake.jpg
```

You can also dynamically resize and scale images with our Responsive Breakpoints Generator.
https://www.responsivebreakpoints.com/

Transforming Content for Optimization

- Resizing
- **Cropping**
- Gravity
- Format
- Quality

Cropping

Decide how to crop the image to fit into the requested height and width, with many different options.

```
cloudinary.url("cheesecake.jpg", {transformation: {width: 300, height: 300, crop: "pad"}})

http://res.cloudinary.com/dannyv-training/image/upload/c_pad,h_300,w_300/cheesecake.jpg
```

Transforming Content for Optimization

- Resizing
- Cropping
- **Gravity**
- Format
- Quality

Gravity

Specify a location in an image or video that is used as the focus for another transformation.

Original

w_300,h_300,c_crop

w_300,h_300,c_thumb, g_auto

```
cloudinary.url("dog.jpg", {transformation: {width: 300, height: 300, crop: "thumb", gravity: "auto"}})

http://res.cloudinary.com/dannyv-training/image/upload/c_thumb,g_auto,h_300,w_300/dog.jpg
```

Transforming Content for Optimization

- Resizing
- Cropping
- Gravity
- **Format**
- Quality

Format

Convert assets to other formats for displaying in your web site or application (output formats).

- Specify image and video format, e.g. on native mobile, based on device capabilities.
- Allow Cloudinary to deliver the optimal format for web delivery scenarios with f_auto for images and video.

```
cloudinary.url("AllyCat.jpg", {transformation: {fetch_format: "auto"}})

http://res.cloudinary.com/dannyv-training/image/upload/f_auto/AllyCat.jpg
```

Transforming Content for Optimization

- Resizing
- Cropping
- Gravity
- **Format**
- Quality

Format

#cfd1d5	#222126	#857b70	#a9a9a8	#5e544c
#696d74	#575d70	#494d58	#969aa5	#ba9f62

URL	https://res.cloudinary.com/dannyv-training/image/upload/AllyCat.jpg
Remote IP	151.101.61.137
Method	GET
HTTP Status	HTTP/2 200
Original Dimensions	2000 x 1500px | Aspect Ratio: 4:3
Display Dimensions	653 x 489px ⇩ | Aspect Ratio: 653:489 | Scale Ratio: 3.06:3.07
server	Cloudinary
content-type	JPEG Image
content-length	216.19 KB (221376 B)
server-timing	fastly Duration 1 ms start 2021-12-16T13:06:21.602Z

#cfd1d5	#222226	#857b70	#a9a9a7	#5d544c
#696d71	#585d70	#494d57	#9799a5	#b49486

URL	https://res.cloudinary.com/dannyv-training/image/upload/f_auto/AllyCat.jpg
Remote IP	151.101.61.137
Method	GET
HTTP Status	HTTP/2 200
Original Dimensions	2000 x 1500px | Aspect Ratio: 4:3
Display Dimensions	653 x 489px ⇩ | Aspect Ratio: 653:489 | Scale Ratio: 3.06:3.07
server	Cloudinary
content-type	WEBP Image
content-length	117.11 KB (119916 B)
server-timing	fastly Duration 1 ms start 2021-12-16T13:06:24.211Z

Cloudinary

Transforming Content for Optimization

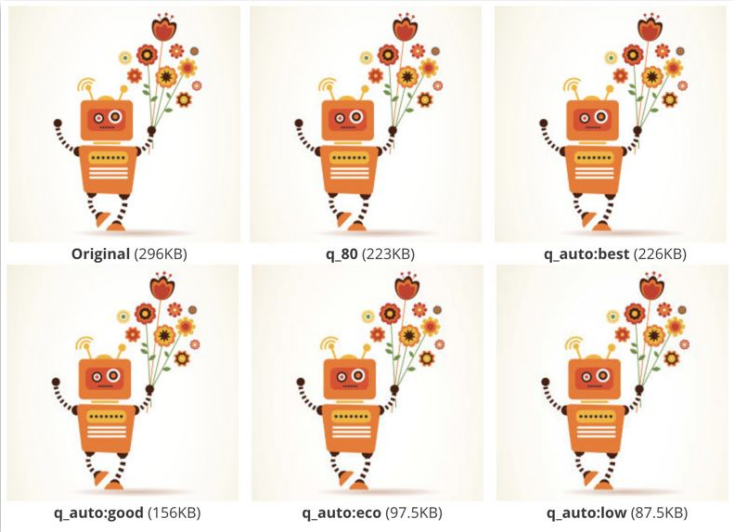
- Resizing
- Cropping
- Gravity
- Format
- **Quality**

Quality

Control the visual quality and compression level of assets.

Allow Cloudinary to deliver the optimal quality for each viewing device with q_auto for images and video.

```
cloudinary.url("robot.jpg",
{transformation: {quality: "auto"}})

http://res.cloudinary.com/dannyv-training/image/upload/q_auto/robot.jpg
```



Transforming Content for Aesthetics

- Rounding
- Borders
- Background Color
- Effects and Filters
- Overlays

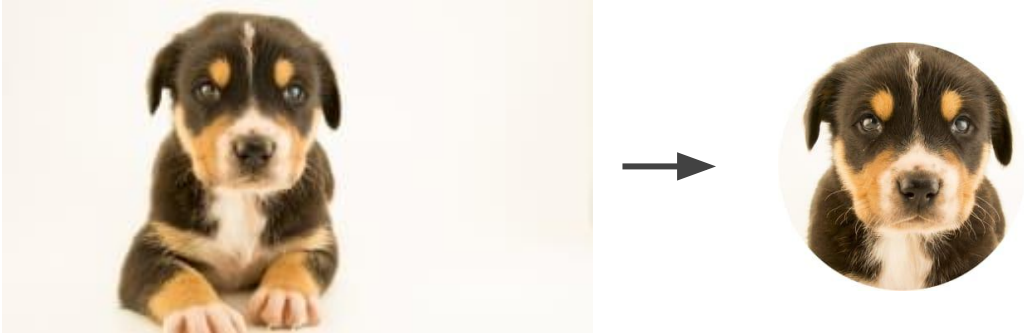
Please run `node transforming_prerequisites.js` before trying the examples found in `transforming.js`

Transforming Content for Aesthetics

- **Rounding**
- Borders
- Background Color
- Effects and Filters
  - Color
  - Improve
  - Artistic Filters
- Overlays

Rounding

Create circular images, being able to manipulate each corner.



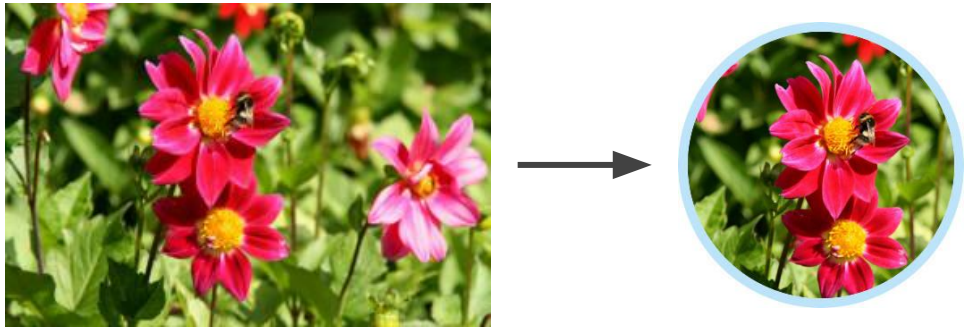
```
cloudinary.url("dog.jpg", {transformation:
   {width: 300, height: 300, crop: "thumb", radius: "max",
   gravity: "auto", fetch_format: "auto", quality: "auto"}})

http://res.cloudinary.com/dannyv-training/image/upload/c_thumb,f_auto,g_auto,h_300,q_auto,r_max,w_300/dog.jpg
```

Transforming Content for Aesthetics

- Rounding
- **Borders**
- Background Color
- Effects and Filters
  - Color
  - Improve
  - Artistic Filters
- Overlays

Borders

Add solid borders around your content with named or RGB hex colors.



```
cloudinary.url("sample.jpg", {transformation:
   {width: 300, height: 300, crop: "thumb",
   gravity: "auto", fetch_format: "auto", quality: "auto", radius: "max",
   border: "10px_solid_rgb:bde4fb"}})

http://res.cloudinary.com/dannyv-training/image/upload/bo_10px_solid_rgb:bde4fb
,c_thumb,f_auto,g_auto,h_300,q_auto,r_max,w_300/sample.jpg
```

Transforming Content for Aesthetics

- Rounding
- Borders
- **Background Color**
- Effects and Filters
  - Color
  - Improve
  - Artistic Filters
- Overlays

Background Color

Pad your asset with a chosen background color.

Allow Cloudinary to automatically set the background color to the most prominent color in the asset with b_auto.

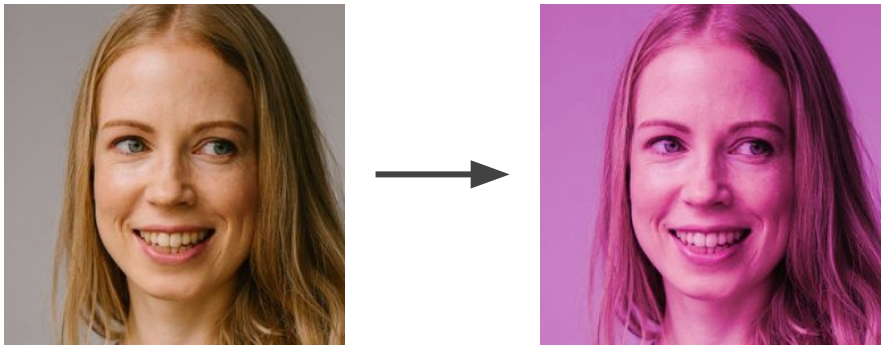
```
cloudinary.url("face.jpg",
{transformation: {
   width: 300, height: 300, crop:
"pad", fetch_format: "auto",
quality: "auto", background:
"auto"}})

http://res.cloudinary.com/dannyv-tr
aining/image/upload/bo_10px_solid_r
gb:bde4fb,c_thumb,f_auto,g_auto,h_3
00,q_auto,r_max,w_300/sample.jpg
```

Transforming Content for Aesthetics

- Rounding
- Borders
- Background Color
- **Effects and Filters**
  - **Color**
  - Improve
  - Artistic Filters
- Overlays

Effects and Filters - Color

Changing the intensities of colors, correcting color imbalance, applying colorization filters, and removing or replacing colors.



```
cloudinary.url("face.jpg", {transformation: {width: 300, height: 300,
   crop: "thumb", gravity: "face", fetch_format: "auto", quality: "auto",
   effect: "tint:40:magenta"}})
```

```
http://res.cloudinary.com/dannyv-training/image/upload/c_thumb,e_tint:
40:magenta,f_auto,g_face,h_300,q_auto,w_300/face.jpg
```

Transforming Content for Aesthetics

- Rounding
- Borders
- Background Color
- **Effects and Filters**
  - Color
  - **Improve**
  - Artistic Filters
- Overlays

Cloudinary

Effects and Filters - Improve

Change an asset's visual appearance with a large number of available effects and filters.

Allow Cloudinary to automatically adjust colors, contrast and lightness with e_improve.

```
cloudinary.url("lake.jpg", {transformation: {effect: "improve:outdoor"}})

http://res.cloudinary.com/dannyv-training/image/upload/e_improve:outdoor/lake.jpg
```

Transforming Content for Aesthetics

- Rounding
- Borders
- Background Color
- **Effects and Filters**
  - Color
  - Improve
  - **Artistic Filters**
- Overlays

Cloudinary

Effects and Filters - Artistic Filters

Change an asset's visual appearance with a large number of available effects and filters.

Brighten highlights, intensify shadows, apply sepia-like filters, add vignetting and more with our `art:<filter>` effects.

```
cloudinary.url("lake.jpg", {transformation: {effect: "art:zorro"}})

http://res.cloudinary.com/dannyv-training/image/upload/e_art:zorro/lake.jpg
```

Transforming Content for Aesthetics

- Rounding
- Borders
- Background Color
- Effects and Filters
  - Color
  - Improve
  - Artistic Filters
- **Overlays**

Overlays

Add text or another asset over the base asset.


```
cloudinary.url("cheesecake.jpg",
{transformation: [{width: 300, height:
300, crop: "thumb", gravity: "auto"},
{overlay: {font_family: "Arial",
font_size: 30, text: "Delicious!"},
color: "lime", height: "30", gravity:
"south_west", x: 5, y: 5}]})
```


```
cloudinary.url("dog.mp4",
{resource_type: "video",
transformation: [{width: 400, crop:
"scale"},
{overlay: "cloudinary_logo", height:
"50", gravity: "south_east", x: 10, y:
10}]})
```

Presets and Named Transformations

- Upload Presets
  - **Via the DAM (Digital Asset Management System)**
- Named Transformations

Examples found in `presets.js`

Creating Presets

- **Upload Presets**
  - Via DAM
- Named Transformations

Upload Presets

Define a group of actions to be applied when uploading an asset, such as an image or video.

Unsigned example for front-end widgets and API calls.

```
cloudinary.api.create_upload_preset(
   { name: "unsigned-image", unsigned: true,
      tags: "unsigned", allowed_formats: "jpg,png" })
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Signed example for backend scripts with access to API_SECRET credentials.

```
cloudinary.api.create_upload_preset(
   { name: "signed-image", unsigned: false,
      tags: "signed",allowed_formats: "jpg,png" })
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

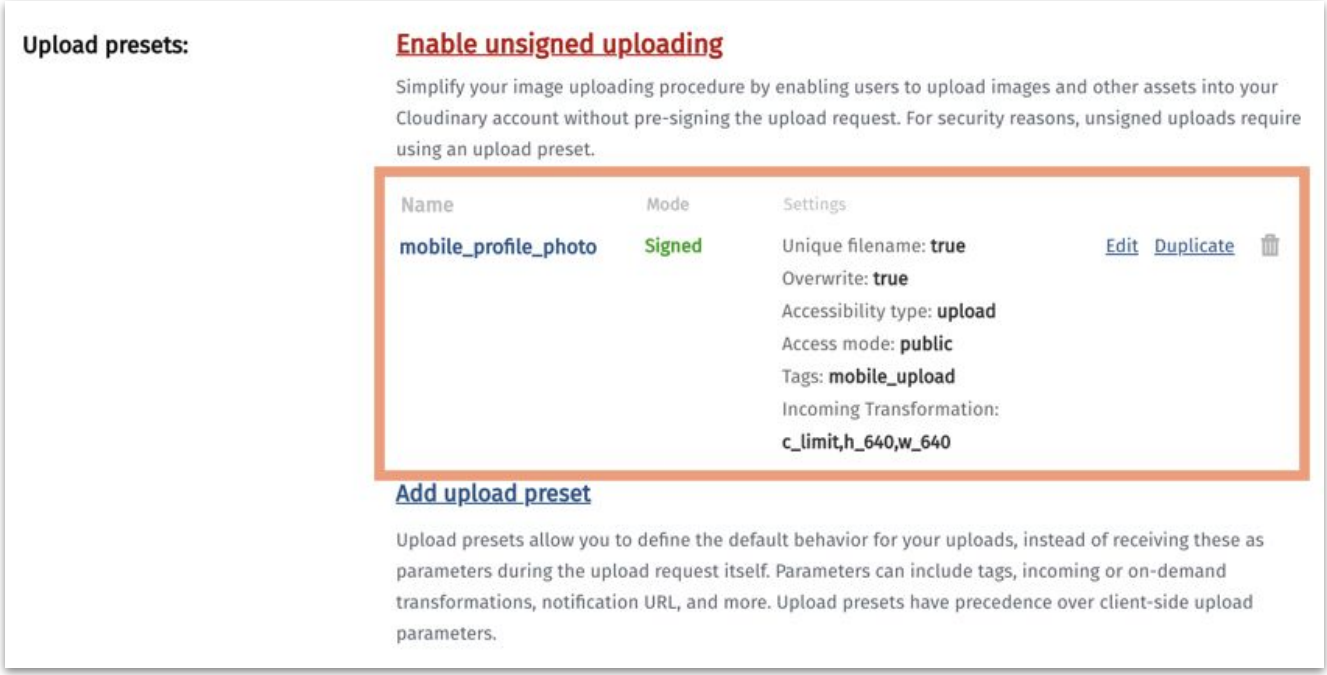
Creating Presets

- **Upload Presets**
  - **Via DAM**
- Named Transformations

Upload Presets - via Digital Asset Management System

Can define an Upload Preset via the DAM, as well as the Admin API.

Only signed Upload Presets can be applied via the DAM for uploads.



Creating Presets

- Upload Presets
  - Via DAM
- **Named Transformations**

Named Transformations

Take one or more transformations you have created and develop a codename for them.

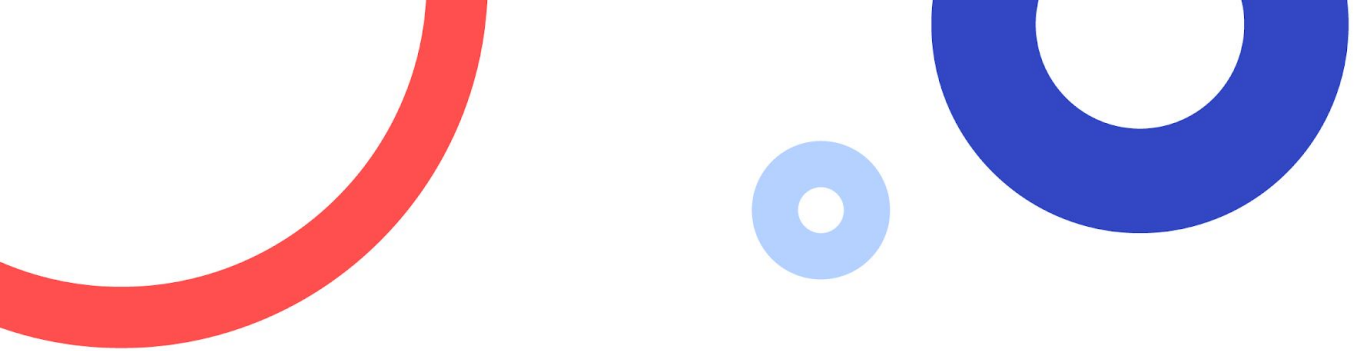
Example:

- Creates a transformation named “standard”
- When applied to an asset, it transforms it on-the-fly to 150x150 with the thumb crop style and our auto-gravity transformation.

```
cloudinary.api.create_transformation("standard", "w_150,h_150,c_thumb,g_auto")
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Applied as a transformation on an image

```
cloudinary.url("cheesecake.jpg", {transformation:
["standard"]})

http://res.cloudinary.com/dannyv-training/image/upload/t
_standard/cheesecake.jpg
```



Setting Up Backups

- Enable
- Using Storage Buckets
- Override Settings

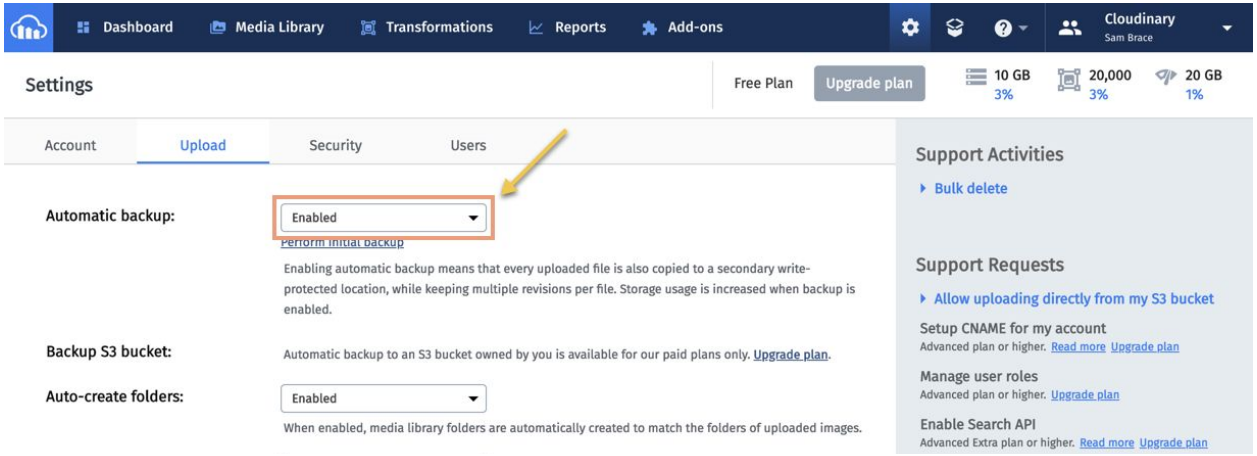
Examples found in `backup.js`

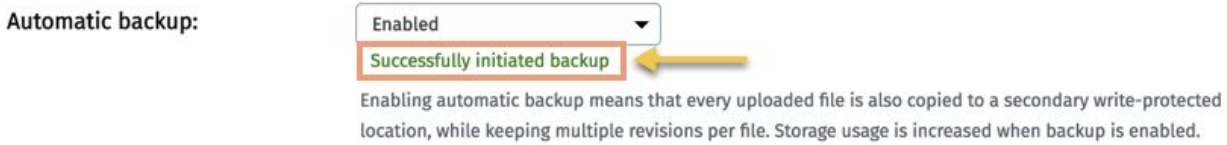
Setting Up Backups

- **Enable**
- Using Storage Buckets
- Override Settings

Enable

Keep revisions of your existing assets, as well as deleted files that you can restore.





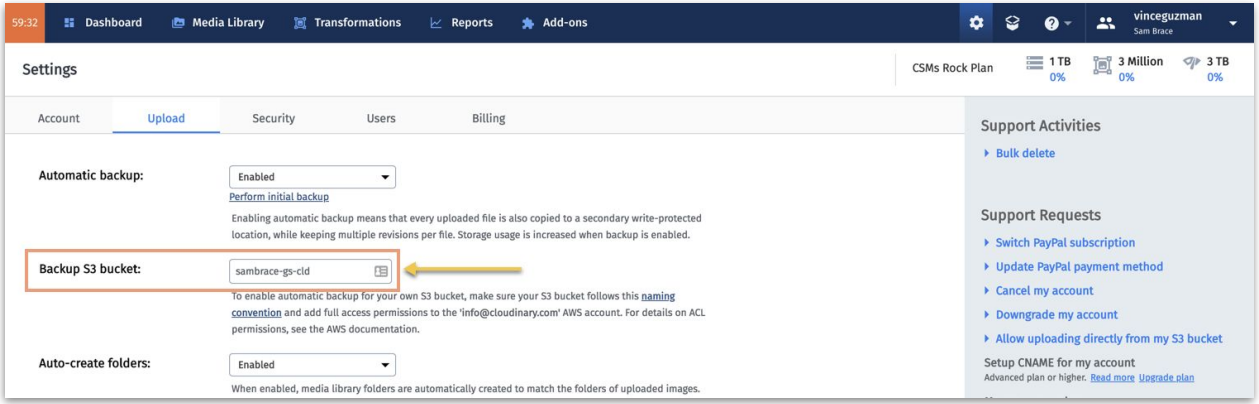
Setting Up Backups

- Enable
- **Using Storage Buckets**
- Override Settings

Using Storage Buckets

Set a designated bucket or secondary location to keep your files.

We support backups through Amazon Simple Storage Service (e.g. Amazon S3) and Google Cloud Storage.



Setting Up Backups

- Enable
- Using Storage Buckets
- **Override Settings**

Override Settings

Specify the backup boolean parameter in the upload API call.

With the global backup setting not enabled, adding the backup parameter and setting it to “true” when uploading the file sample.jpg, will ensure that the file is backed-up.

```
cloudinary.uploader.upload("./assets/cat.jpg", {backup: true})
   .then(result => {console.log(result)})
   .catch(error => {console.log(error)});
```

Fine Tuning Your Account Settings

- Review Your Reports
  - **Error Reports**
- Review Our Add-Ons

Fine Tuning Your Account Settings

- **Review Your Reports**
  - Error Reports
- Review Our Add-Ons

Review Your Reports

Periodically review your account's Reports section in the DAM for some helpful optimization tips.

Consider using automatic WebP, JPEG-XR and JPEG image format selection for each different browser using 'f_auto' in delivery URLs.

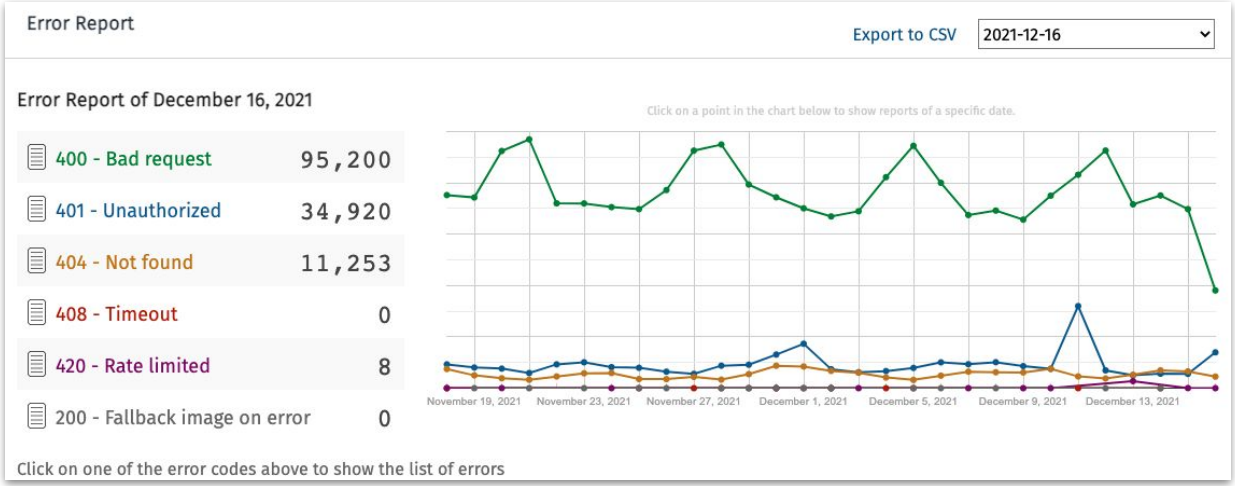
Fine Tuning Your Account Settings

- Review Your Reports
  - **Error Reports**
- Review Our Add-Ons

Review Your Reports - Error Reports

If you experience any resource delivery issues, your Error Report page can be quite useful.

Requests made to your Cloudinary account via delivery URLs or API calls are processed and any errors generated are displayed in an interactive web page in Cloudinary's Management Console.

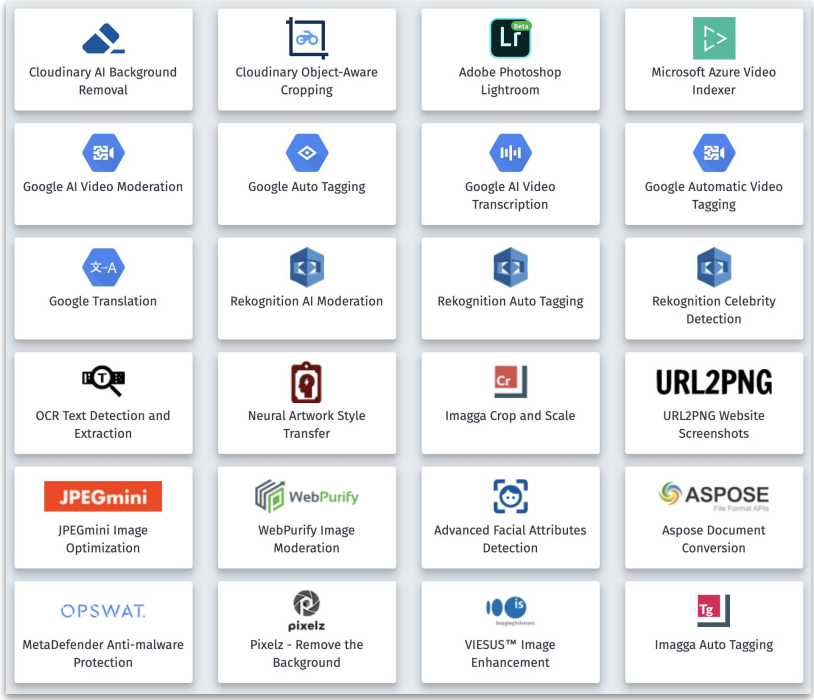


Fine Tuning Your Account Settings

- Review Your Reports
  - Error Reports
- **Review Our Add-Ons**

Review Our Add-Ons

Inspect your account's Add-Ons section in the DAM to determine if any could be helpful for your workflow.



Next Steps for Further Support

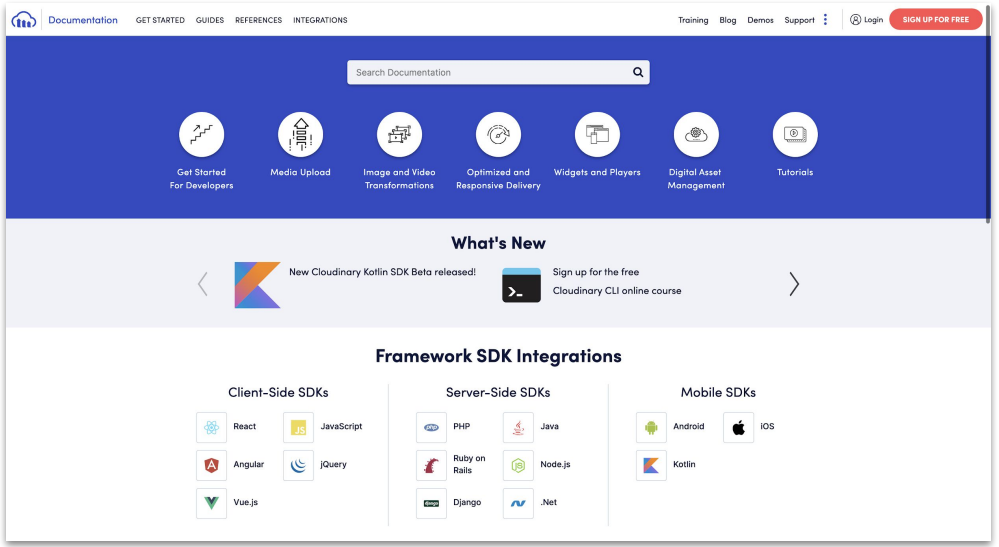
- Read Our Documentation
- Join Our Community
- Review GitHub Repositories
- Subscribe to Status Page
- Talk to Us
- Take Another Class!

Next Steps for Further Support

- **Read Our Documentation**
- Join Our Community
- Review GitHub Repositories
- Subscribe to Status Page
- Talk to Us
- Take Another Class!

Read Our Documentation

We offer a range of guides that describe available features and present common use-cases, including detailed code examples where relevant for all of our SDKs.



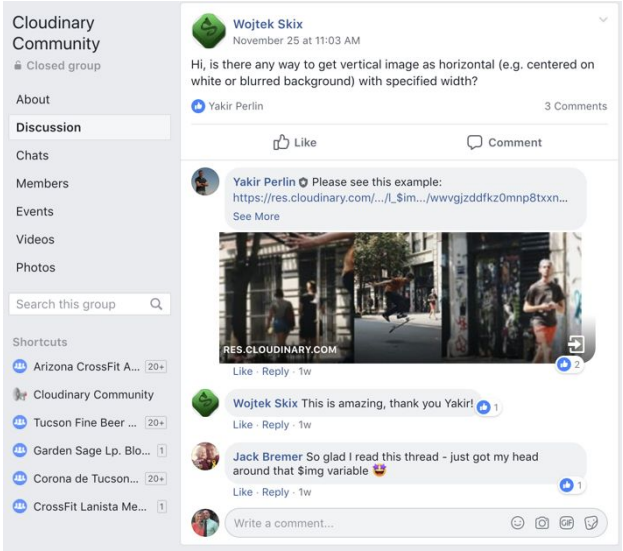
https://cloudinary.com/documentation/

Next Steps for Further Support

- Read Our Documentation
- **Join Our Community**
- Review GitHub Repositories
- Subscribe to Status Page
- Talk to Us
- Take Another Class!

Join Our Community

Request to join our Facebook group devoted to peer-to-peer conversations about the service.



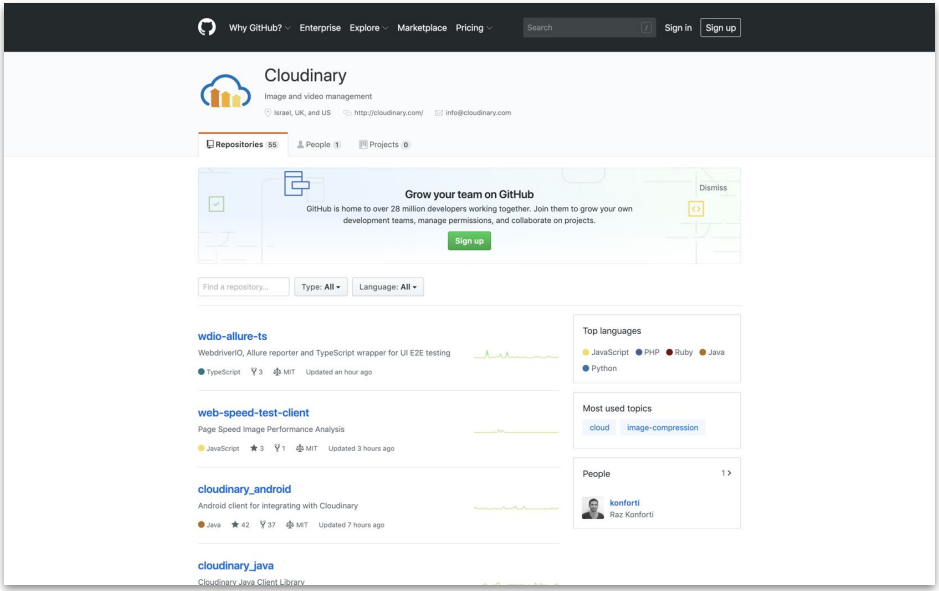
https://www.facebook.com/groups/CloudinaryCommunity/

Next Steps for Further Support

- Read Our Documentation
- Join Our Community
- **Review GitHub Repositories**
- Subscribe to Status Page
- Talk to Us
- Take Another Class!

Review GitHub Repositories

Access our sample projects to help you with building your own work with Cloudinary.



https://github.com/cloudinary

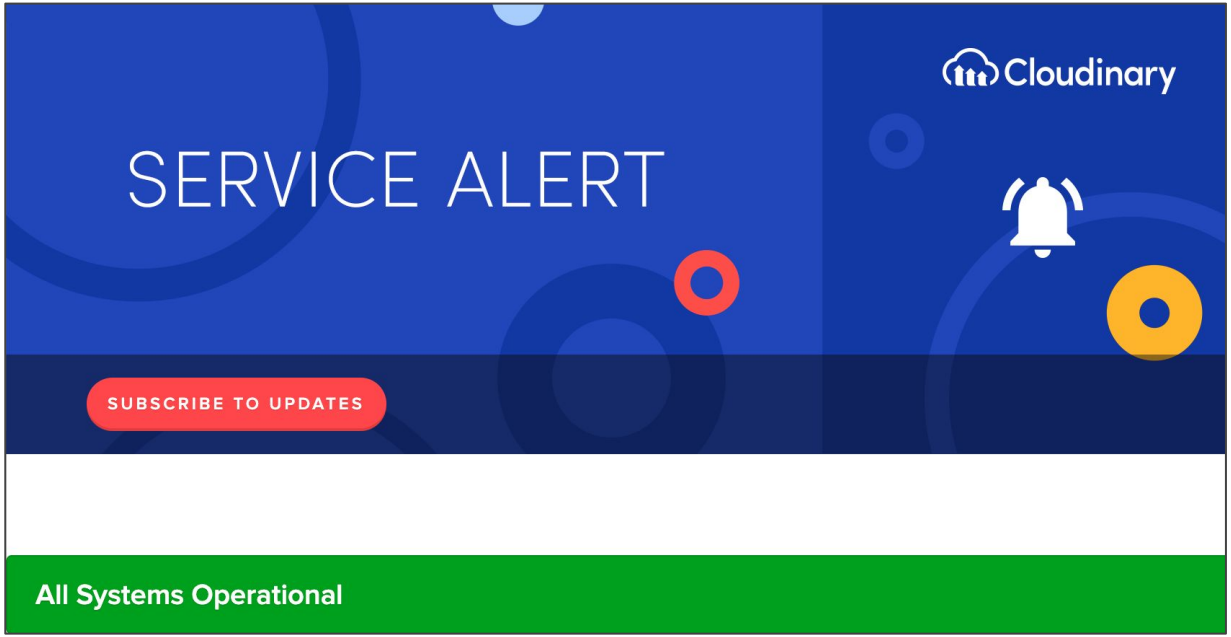
Next Steps for Further Support

- Read Our Documentation
- Join Our Community
- Review GitHub Repositories
- **Subscribe to Status Page**
- Talk to Us
- Take Another Class!

Subscribe to Status Page

Keep up-to-date with any maintenance work via our Status page.

You can subscribe to receive updates via email and/or SMS.



https://status.cloudinary.com/

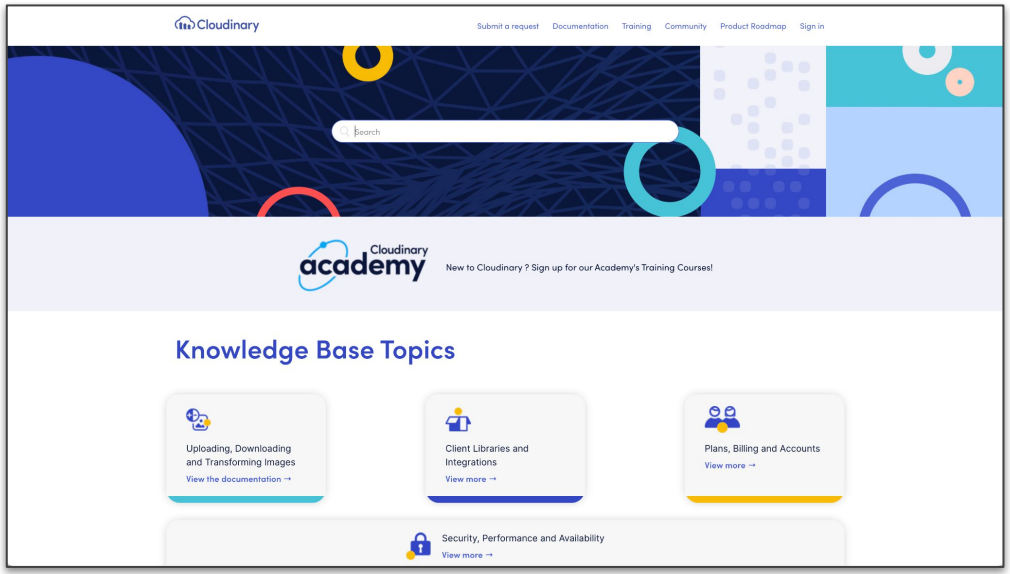
Next Steps for Further Support

- Read Our Documentation
- Join Our Community
- Review GitHub Repositories
- Subscribe to Status Page
- **Talk to Us**
- Take Another Class!

Cloudinary

Talk to Us

We are always happy to answer your questions, as we have dedicated support staff for our developer community.



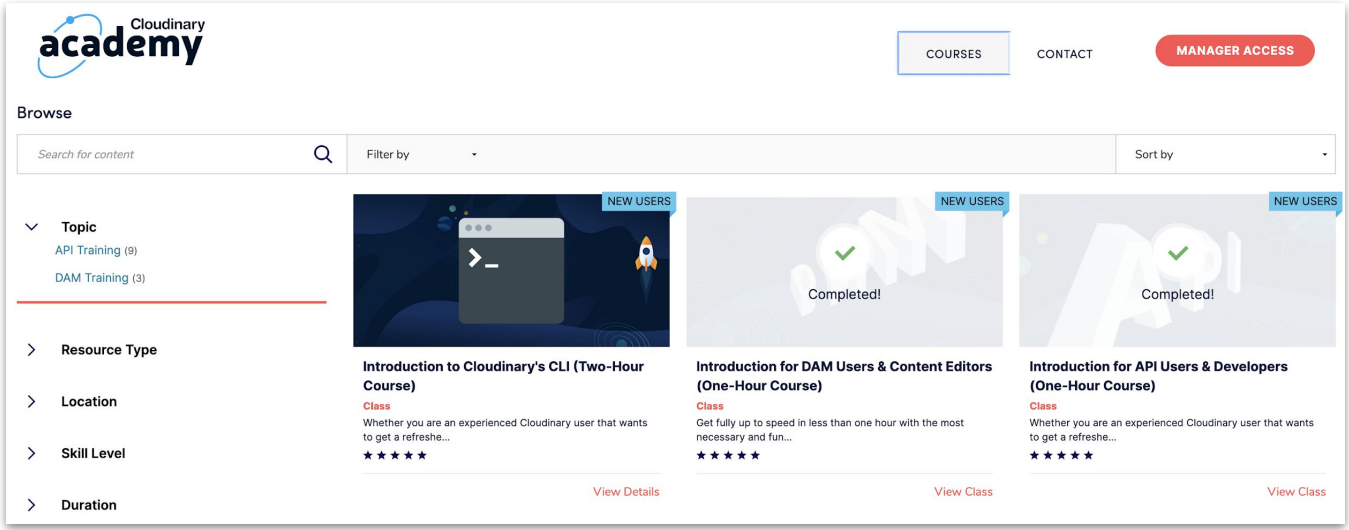
https://support.cloudinary.com/hc/en-us/requests/new

Next Steps for Further Support

- Read Our Documentation
- Join Our Community
- Review GitHub Repositories
- Subscribe to Status Page
- Talk to Us
- **Take Another Class!**

Take Another Class!

We have plenty of in-depth courses in the Cloudinary Academy, including live classes taught by Cloudinary's brightest team members.



https://training.cloudinary.com/

Cloudinary

Thank You.