



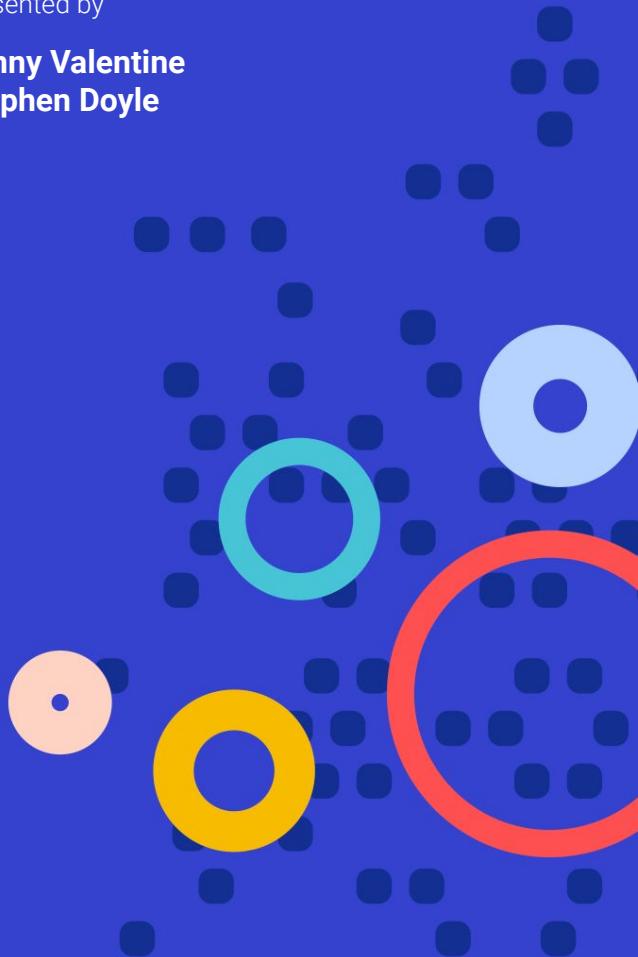
Presented by

Danny Valentine

Stephen Doyle

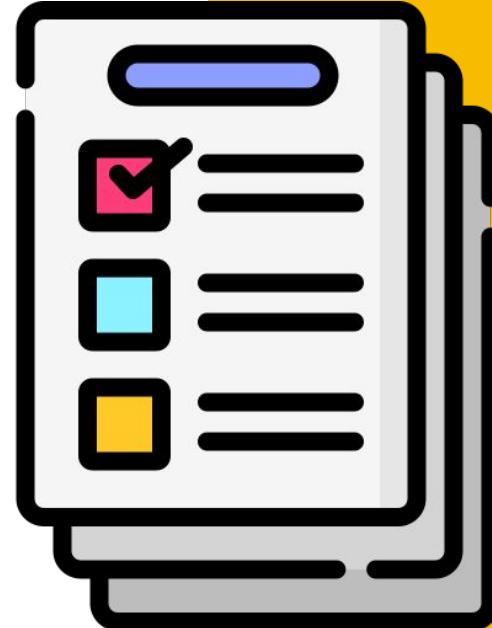
# Introduction to Cloudinary for API Users and Node.js Developers

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

We recommend using HomeBrew:

```
$ brew install node
```

<https://nodejs.org/tr/download/package-manager/#macos>

### Windows Users

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

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

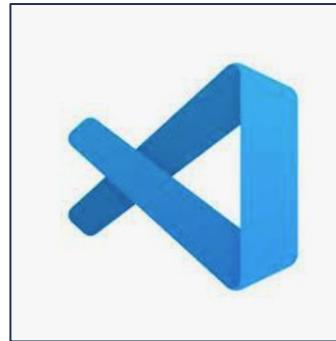
$ npm --version
8.1.0
```

# 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`

### Dashboard

#### Account Details

Cloud name: dtmvrbj50 [Copied to clipboard](#)

API Key: 148629267839852 [Copied to clipboard](#)

API Secret: \*\*\*\* [Copy to clipboard](#) [Reveal](#)

API Environment variable: `CLOUDINARY_URL=cloudinary://*****:*****@*****` [Copied to clipboard](#)

# Environment Setup

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

## 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

## 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.

<https://res.cloudinary.com/demo/image/upload/sample.jpg>

Examples found in `upload.js`

# Uploading Basics

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

## 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: 'fbb0f017dba3395f026e91f23f9fb2f',  
  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
```

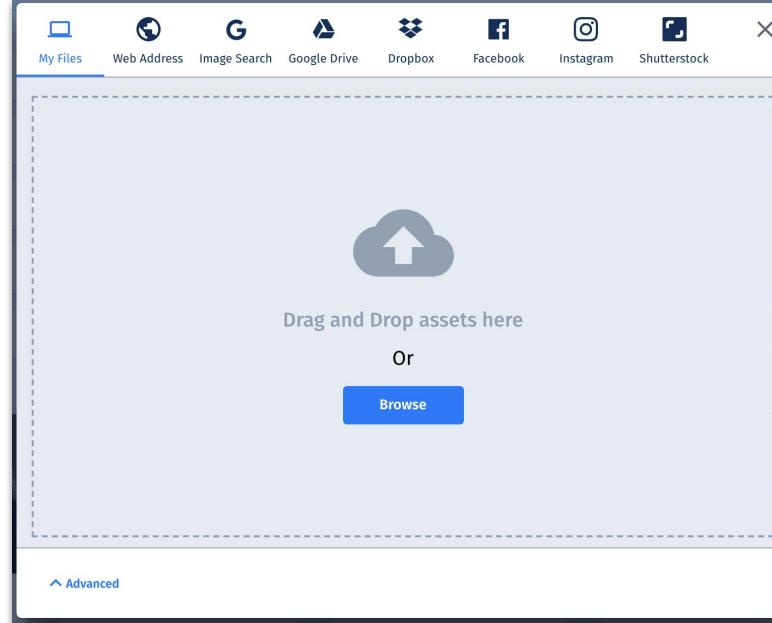
# Uploading Basics

- 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

## 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/demo/image/upload/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

## 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



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](https://res.cloudinary.com/sambrace/image/upload/v1583856029/cute_animals/kitten.jpg)

### Cool Tip!

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

# Uploading Basics

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

## 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)

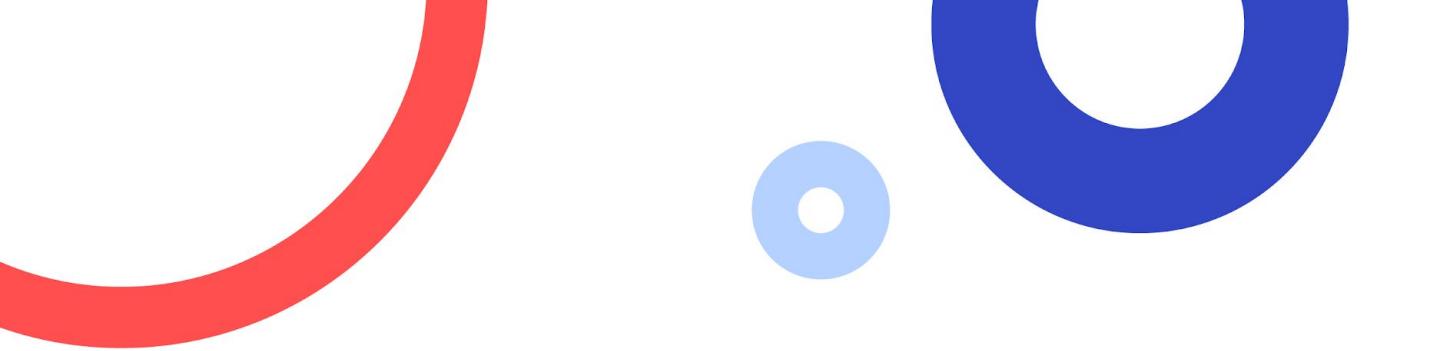
```
cloudinary.uploader.upload("s3://my-bucket/my-path/example.jpg")
  .then(result => {console.log(result)})
  .catch(error => {console.log(error)});
```

### Base64 Data URI

```
cloudinary.uploader.upload("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCQAAAC1HAwCAAAAC01EQVQYV2NgYAAAAAMAAWgmWQ0AAAASUVORK5CYII=")
  .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

# 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
```

Auto upload mapping:

Folder:  →

URL prefix:

<https://res.cloudinary.com/dannyyv-training/image/upload/remote-media/sample.jpg> →  
<https://cloudinary-training.github.io/cld-advanced-concepts/assets/sample.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.

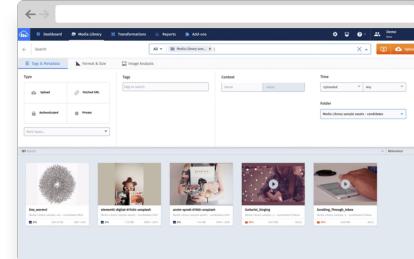
### List all assets

```
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.



# Managing Uploaded Content

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

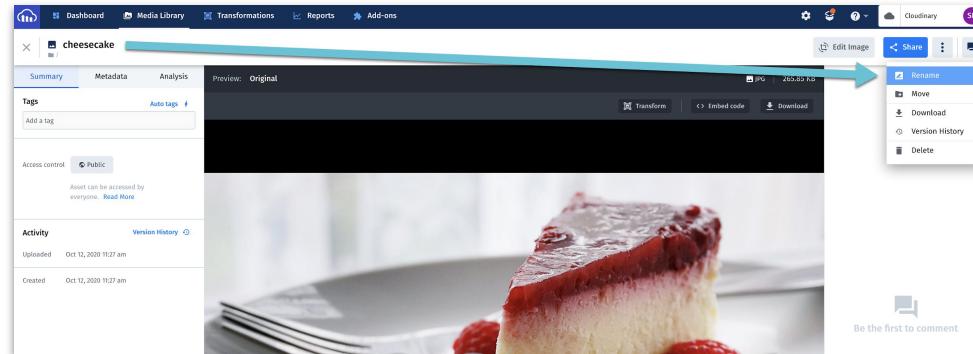
## Rename



● 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)});
```



# 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.

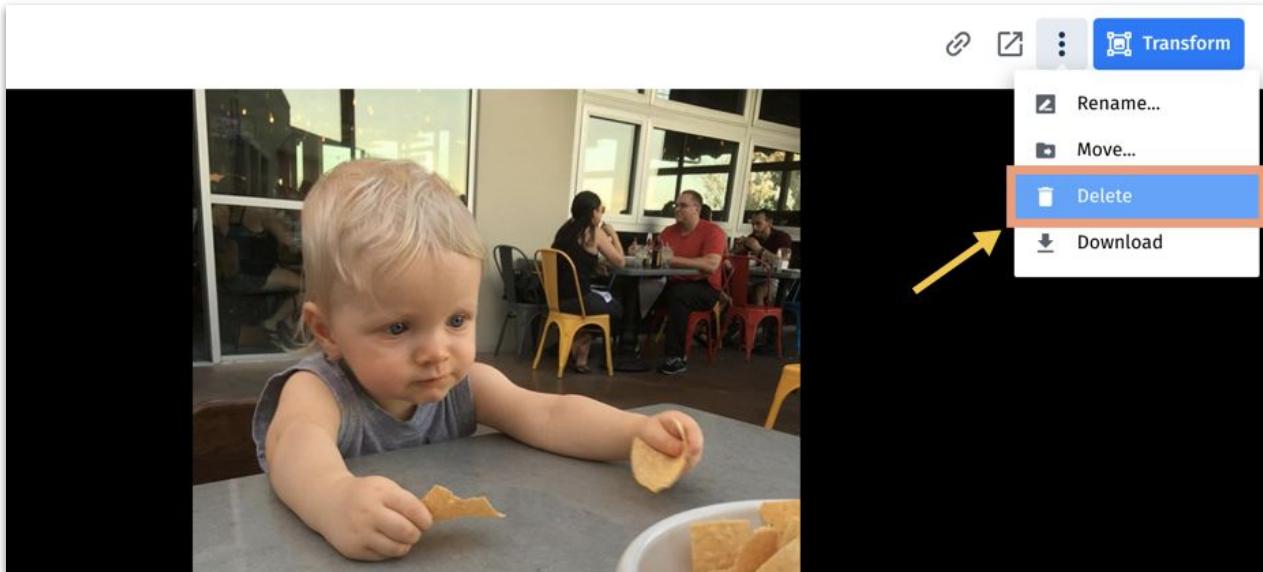
```
cloudinary.api.delete_resources(["squirrel", "sheep"])
  .then(result => {console.log(result)})
  .catch(error => {console.log(error)});
```

# 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

## 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.

```
clouddinary.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/>



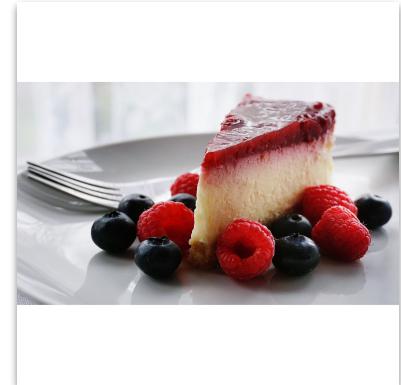
# Transforming Content for Optimization

- 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/dannyyv-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.

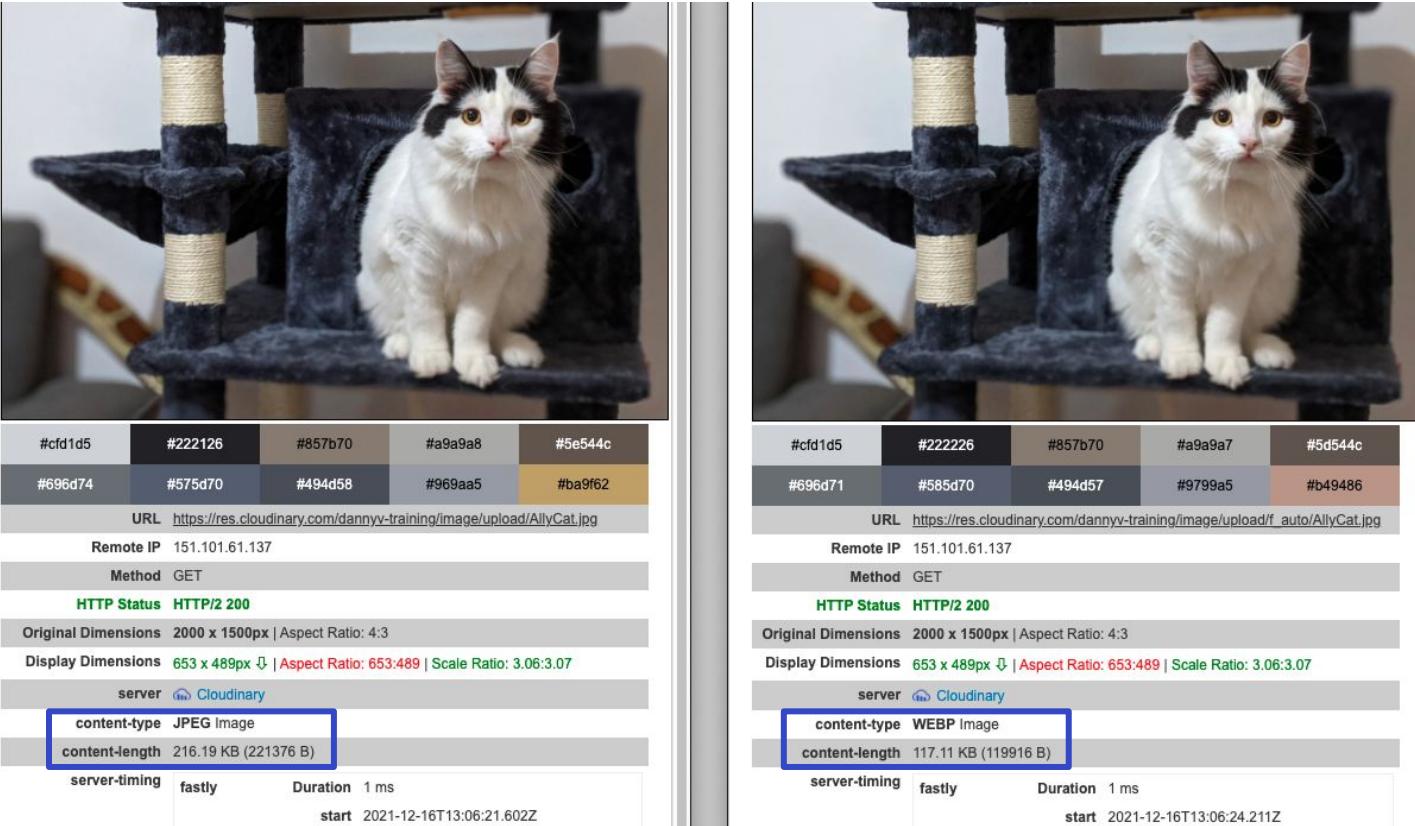
```
cloudinary.url("AllyCat.jpg", {transformation: {fetch_format: "auto"}})
```

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

# Transforming Content for Optimization

- Resizing
- Cropping
- Gravity
- Format
- Quality

## Format



# Transforming Content for Optimization

- Resizing
- Cropping
- Gravity
- Format
- Quality

## Quality

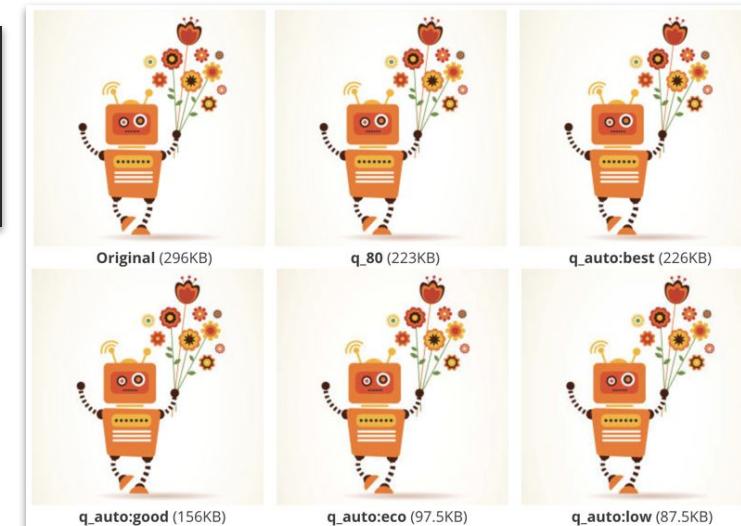


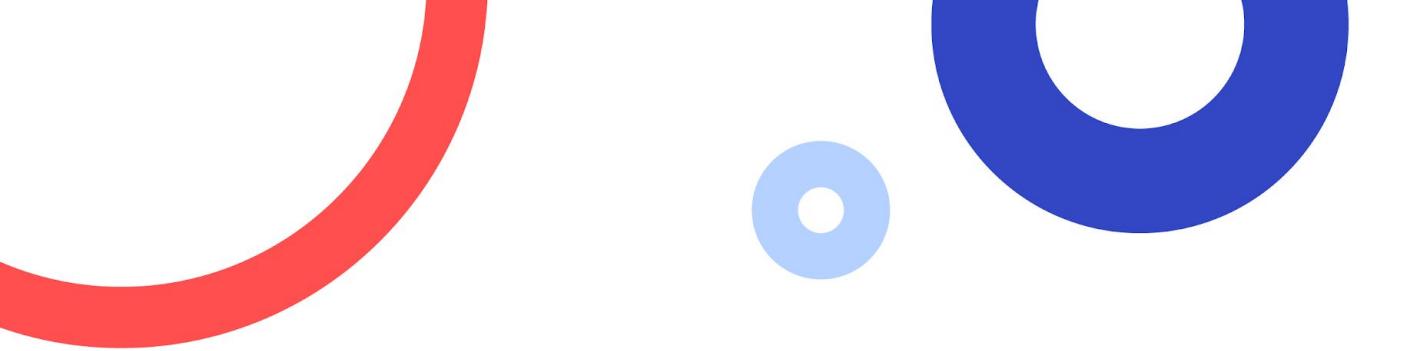
Control the visual quality and compression level of assets.

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/b0_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-trainin/image/upload/b_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

## 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

## 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

## 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

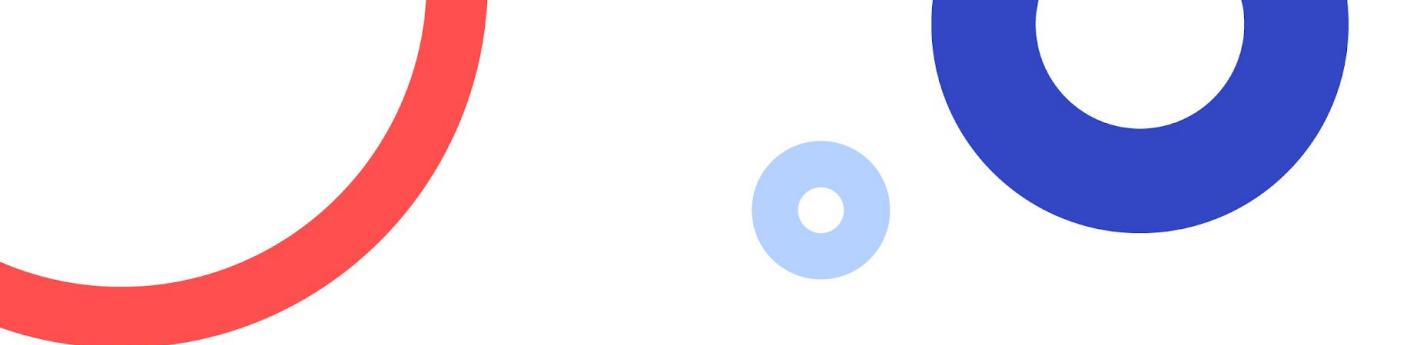


Add text or another asset over the base asset.



```
clouddinary.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}]}))
```

```
clouddinary.url("dog.mp4",
  {resource_type: "video",
   transformation: [{width: 400, crop: "scale"}, {overlay: "clouddinary_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.*

### Upload presets:

#### Enable unsigned uploading

Simplify your image uploading procedure by enabling users to upload images and other assets into your Cloudinary account without pre-signing the upload request. For security reasons, unsigned uploads require using an upload preset.

Name	Mode	Settings	Edit	Duplicate	Delete
mobile_profile_photo	Signed	Unique filename: true Overwrite: <b>true</b> Accessibility type: <b>upload</b> Access mode: <b>public</b> Tags: <b>mobile_upload</b> Incoming Transformation: <b>c_limit,h_640,w_640</b>	<a href="#">Edit</a>	<a href="#">Duplicate</a>	

#### Add upload preset

Upload presets allow you to define the default behavior for your uploads, instead of receiving these as parameters during the upload request itself. Parameters can include tags, incoming or on-demand transformations, notification URL, and more. Upload presets have precedence over client-side upload parameters.

# 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.

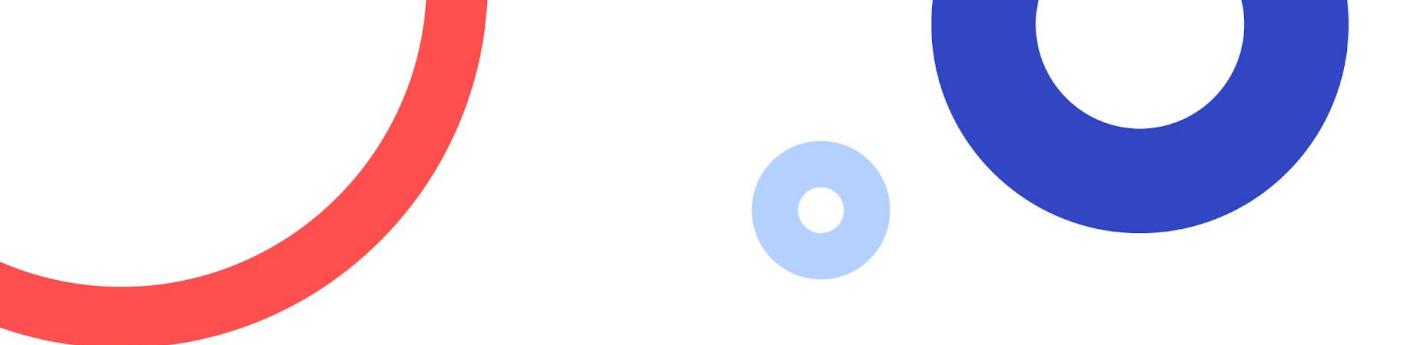
```
cloudinary.api.create_transformation("standard", "w_150,h_150,c_thumb,g_auto")
  .then(result => {console.log(result)})
  .catch(error => {console.log(error)});
```

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.

The screenshot shows the Cloudinary settings page with the 'Upload' tab selected. In the 'Automatic backup:' section, a dropdown menu is open, showing 'Enabled'. Below the dropdown, a message says 'Successfully initiated backup'. To the right, there's a sidebar titled 'Support Activities' and 'Support Requests'.

Settings

Free Plan Upgrade plan

10 GB 3% 20,000 3% 20 GB 1%

Support Activities

Bulk delete

Support Requests

Allow uploading directly from my S3 bucket

Setup CNAME for my account

Advanced plan or higher. [Read more](#) [Upgrade plan](#)

Manage user roles

Advanced plan or higher. [Upgrade plan](#)

Enabled Search API

Advanced Extra plan or higher. [Read more](#) [Upgrade plan](#)

Automatic backup:

Enabled

Successfully initiated backup

Perform initial backup

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.

Backup S3 bucket:

Automatic backup to an S3 bucket owned by you is available for our paid plans only. [Upgrade plan](#).

Auto-create folders:

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

Automatic backup:

Enabled

Successfully initiated backup

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.

# 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.

The screenshot shows the Clodinary settings interface. At the top, there's a navigation bar with links for Dashboard, Media Library, Transformations, Reports, and Add-ons. On the right side of the header, there are account details for 'vinceguzman' and 'Sam Brace', and usage metrics for 'CSMs Rock Plan' (1 TB 0%, 3 Million 0%, 3 TB 0%).

The main area is titled 'Settings' and contains tabs for Account, Upload (which is selected), Security, Users, and Billing. Under the 'Upload' tab, there are several configuration options:

- Automatic backup:** A dropdown menu set to "Enabled". Below it is a button labeled "Perform initial backup".
- Backup S3 bucket:** An input field containing "sambrace-gs-cld". A yellow arrow points to this field from the left.
- Auto-create folders:** A dropdown menu set to "Enabled". Below it is a note: "When enabled, media library folders are automatically created to match the folders of uploaded images."

On the right side of the settings page, there are two sections: "Support Activities" and "Support Requests".

- Support Activities:** Includes a link to "Bulk delete".
- Support Requests:** Includes links to "Switch PayPal subscription", "Update PayPal payment method", "Cancel my account", "Downgrade my account", and "Allow uploading directly from my S3 bucket".

At the bottom right, there's a note about setting up a CNAME: "Setup CNAME for my account Advanced plan or higher. [Read more](#) [Upgrade plan](#)".

# Setting Up Backups

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

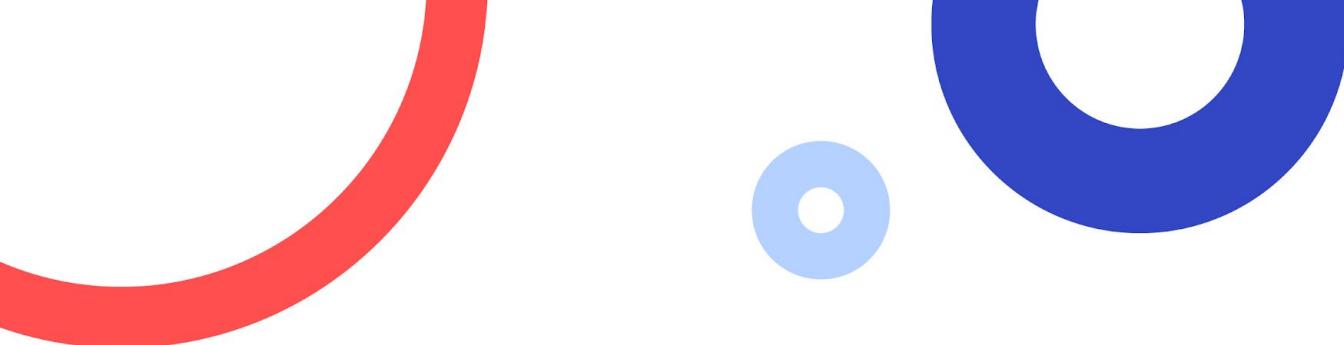
## Override Settings



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.

```
clouddinary.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.

### Usage Report of December 03, 2018

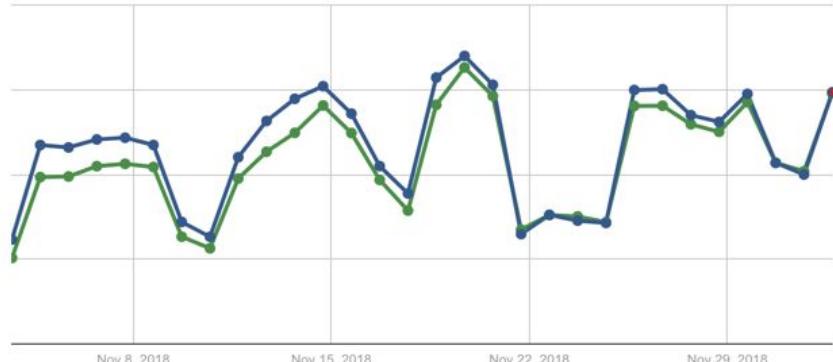
Click on a point in the chart below to show reports of a specific date.

**166 GB**

Bandwidth

**2.96 Million**

Requests



# Fine Tuning Your Account Settings

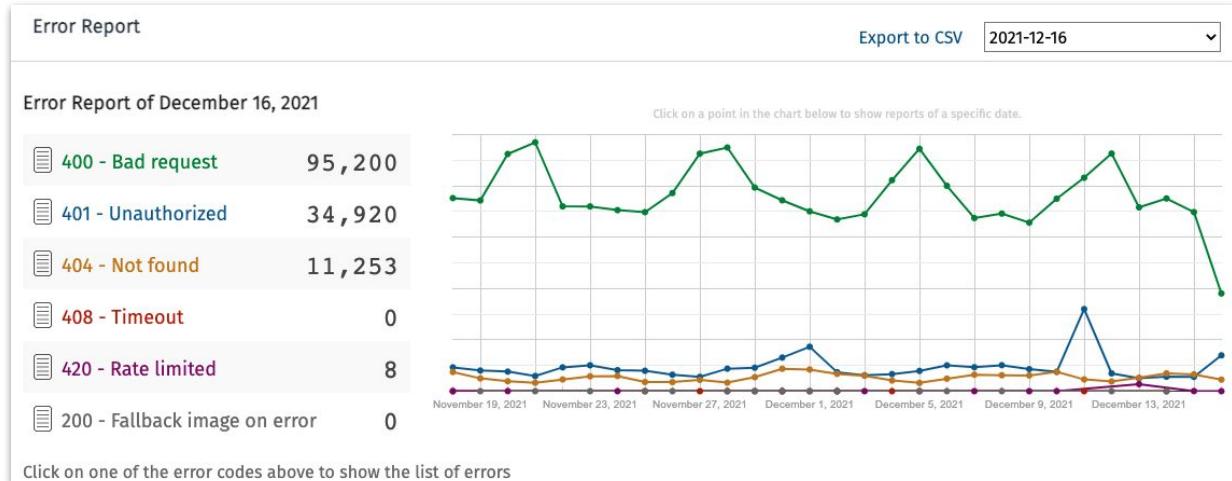
- Review Your Reports
  - Error Reports
- Review Our Add-Ons

## 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.

 Cloudinary AI Background Removal	 Cloudinary Object-Aware Cropping	 Adobe Photoshop Lightroom	 Microsoft Azure Video Indexer
 Google AI Video Moderation	 Google Auto Tagging	 Google AI Video Transcription	 Google Automatic Video Tagging
 Google Translation	 Rekognition AI Moderation	 Rekognition Auto Tagging	 Rekognition Celebrity Detection
 OCR Text Detection and Extraction	 Neural Artwork Style Transfer	 Imagga Crop and Scale	 URL2PNG URL2PNG Website Screenshots
 JPEGmini Image Optimization	 WebPurify Image Moderation	 Advanced Facial Attributes Detection	 Aspose Document Conversion
 OPSWAT MetaDefender Anti-malware Protection	 pixelz Pixelz - Remove the Background	 VIESUST™ Image Enhancement	 Imagga Auto Tagging



## 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.

The screenshot shows the Cloudinary Documentation website. At the top, there's a navigation bar with links for Documentation, GET STARTED, GUIDES, REFERENCES, INTEGRATIONS, Training, Blog, Demos, Support, Login, and SIGN UP FOR FREE. Below the navigation is a search bar labeled "Search Documentation". Underneath the search bar is a row of circular icons representing various documentation categories: Get Started For Developers, Media Upload, Image and Video Transformations, Optimized and Responsive Delivery, Widgets and Players, Digital Asset Management, and Tutorials. Below these icons is a "What's New" section featuring a Kotlin logo and text about the release of the Cloudinary Kotlin SDK Beta. To the right of this section is a link to sign up for the free Cloudinary CLI online course. The main content area is titled "Framework SDK Integrations" and is divided into three columns: Client-Side SDKs (React, Angular, Vue.js, JavaScript), Server-Side SDKs (PHP, Ruby on Rails, Django, Java, Node.js, .Net), and Mobile SDKs (Android, iOS, Kotlin).

<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.

The screenshot shows a Facebook group named "Cloudinary Community". The group is a closed group with 20+ members. A post from user "Wojtek Skix" asks if there's any way to get a vertical image as horizontal. User "Yakir Perlin" replies with a link and a photo example showing a person jumping over a bar. Other users like and comment on the post.

Cloudinary Community  
Closed group

Wojtek Skix November 25 at 11:03 AM  
Hi, is there any way to get vertical image as horizontal (e.g. centered on white or blurred background) with specified width?

Yakir Perlin 3 Comments  
Please see this example:  
[https://res.cloudinary.com/.../l\\_s.../wwwjzddfkz0mp8txn...](https://res.cloudinary.com/.../l_s.../wwwjzddfkz0mp8txn...)  
See More

RES.CLOUDINARY.COM

Like · Reply · 1w

Wojtek Skix This is amazing, thank you Yakir! 1  
Like · Reply · 1w

Jack Bremer So glad I read this thread - just got my head around that \$img variable 😊 1  
Like · Reply · 1w

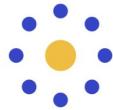
Write a comment...

<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.

The screenshot shows the GitHub repository page for the Cloudinary organization. At the top, there's a banner for GitHub with the text "Grow your team on GitHub" and a "Sign up" button. Below the banner, there are search and filter options: "Find a repository...", "Type: All", and "Language: All". The main area displays four repository cards:

- wdio-allure-ts**: WebdriverIO, Allure reporter and TypeScript wrapper for UI E2E testing. Last updated an hour ago.
- web-speed-test-client**: Page Speed Image Performance Analysis. Last updated 3 hours ago.
- cloudinary\_android**: Android client for integrating with Cloudinary. Last updated 7 hours ago.
- cloudinary\_java**: Cloudinary Java Client Library.

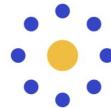
On the right side, there are two boxes: "Top languages" (JavaScript, PHP, Ruby, Java, Python) and "Most used topics" (cloud, image-compression). Below these is a "People" section showing a profile picture and name for "konforti".

<https://github.com/cloudinary>

# Next Steps for Further Support

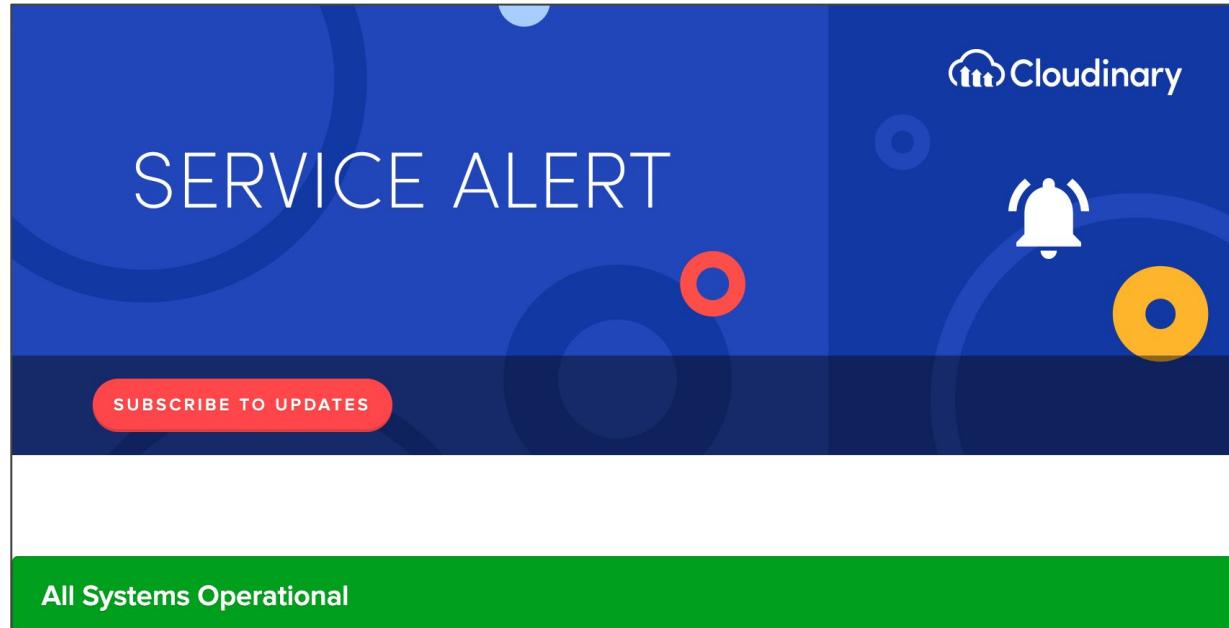
- Read Our Documentation
- Join Our Community
- Review GitHub Repositories
- **Subscribe to Status Page**
- Talk to Us
- Take Another Class!

## 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.

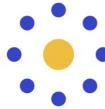


<https://status.cloudinary.com/>

# Next Steps for Further Support

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

## Talk to Us



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

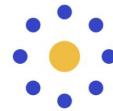
The screenshot shows the Cloudinary support page. At the top, there's a navigation bar with links for 'Submit a request', 'Documentation', 'Training', 'Community', 'Product Roadmap', and 'Sign in'. Below the navigation is a decorative header with abstract shapes in blue, yellow, and teal. A search bar is located in the center of the header. The main content area features the 'Cloudinary Academy' logo and a call to action: 'New to Cloudinary? Sign up for our Academy's Training Courses!'. Below this, there's a section titled 'Knowledge Base Topics' with three cards: 'Uploading, Downloading and Transforming Images' (View the documentation →), 'Client Libraries and Integrations' (View more →), and 'Plans, Billing and Accounts' (View more →). At the bottom, there's a footer section titled 'Security, Performance and Availability' with a 'View more →' link.

<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.

The screenshot shows the Cloudinary Academy website's course browse interface. At the top, there's a navigation bar with the Cloudinary Academy logo, a search bar, filter options, and buttons for 'COURSES', 'CONTACT', and 'MANAGER ACCESS'. On the left, a sidebar offers filters for 'Topic' (API Training and DAM Training), 'Resource Type', 'Location', 'Skill Level', and 'Duration'. The main area displays three course cards:

- Introduction to Cloudinary's CLI (Two-Hour Course)**  
Class  
Whether you are an experienced Cloudinary user that wants to get a refresh...  
★★★★★  
[View Details](#)
- Introduction for DAM Users & Content Editors (One-Hour Course)**  
Class  
Get fully up to speed in less than one hour with the most necessary and fun...  
★★★★★  
[View Class](#)
- Introduction for API Users & Developers (One-Hour Course)**  
Class  
Whether you are an experienced Cloudinary user that wants to get a refresh...  
★★★★★  
[View Class](#)

<https://training.cloudinary.com/>



# Thank You.