Cloudinary academy

Introduction to Cloudinary's CLI

Overview for Developers and Command Line Interface Users

Jen Brissman
Technical Curriculum Engineer

Cloudinary

Topics

- What is a CLI?
- Setting Up Your Environment
- CLI Helper Tools
- Uploading Assets
- Cloudinary URLs
- Managing Assets
- Sync (Local File Management)
- Transforming Assets
- Next Steps for Further Support

What is a CLI?

Command Line Interface

CLI stands for ***Command Line Interface***

- Text-based interface used for interacting with software and computer systems by entering commands into a terminal or console.

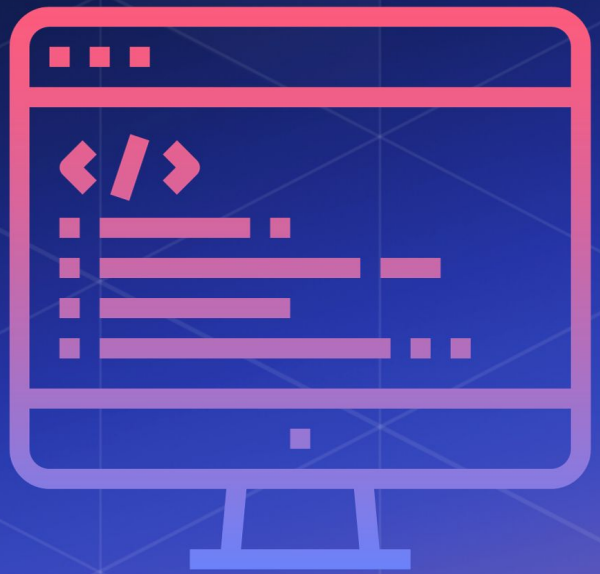
- In a CLI, users interact with the system by typing commands and receiving text-based feedback.

- Alternative to Graphical User Interfaces - GUIs, where users interact with software through graphical elements like windows, icons, and buttons.

Cloudinary CLI

Cloudinary

Setting Up Your Environment



Install Python

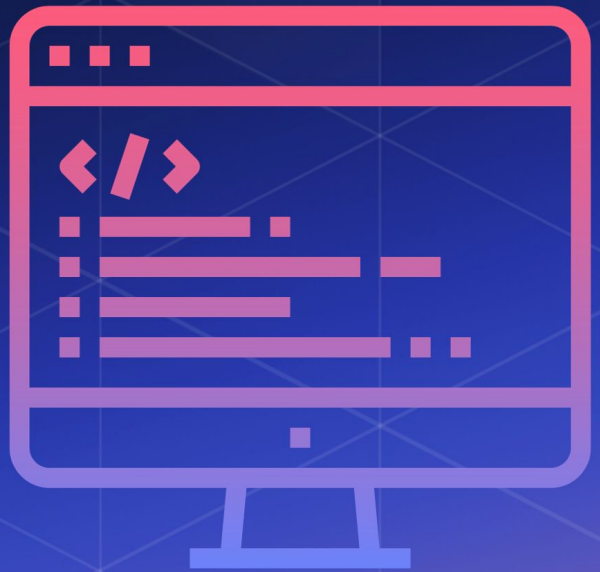
To run Cloudinary's CLI, you need to install Python, **version 3.6 or higher**.

You can download directly from python.org/downloads

Download Python 3.12.0

You can also download and install through Homebrew (Mac)

```
$ brew update && brew upgrade
$ brew install python3
$ python3 --version
> Python 3.9.6
```



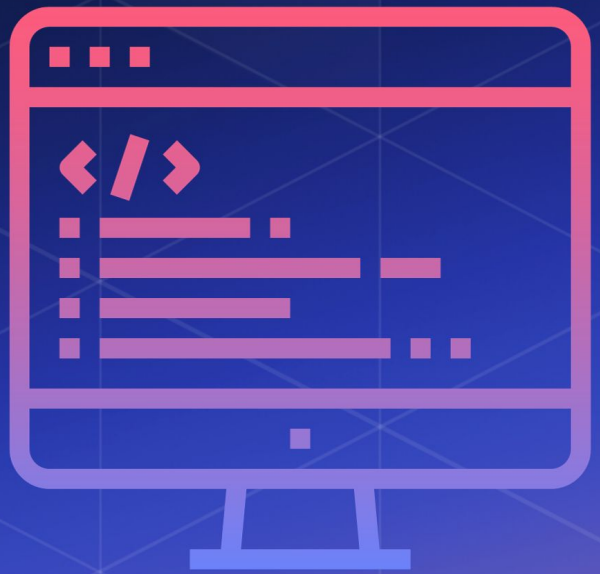
Install Cloudinary

Next, you will need to install the Cloudinary CLI Package.

```
$ pip3 install cloudinary-cli
```

Make sure you have the most updated package.

```
$ pip3 install cloudinary-cli --upgrade
```



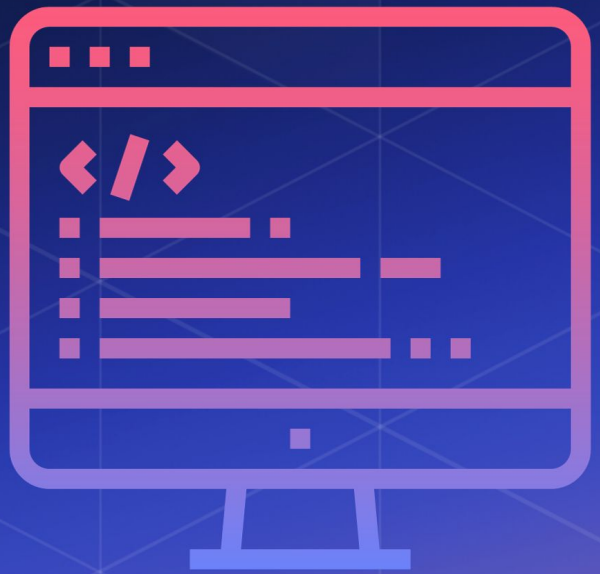
Install Cloudinary

Next, you will need to install the Cloudinary CLI Package.

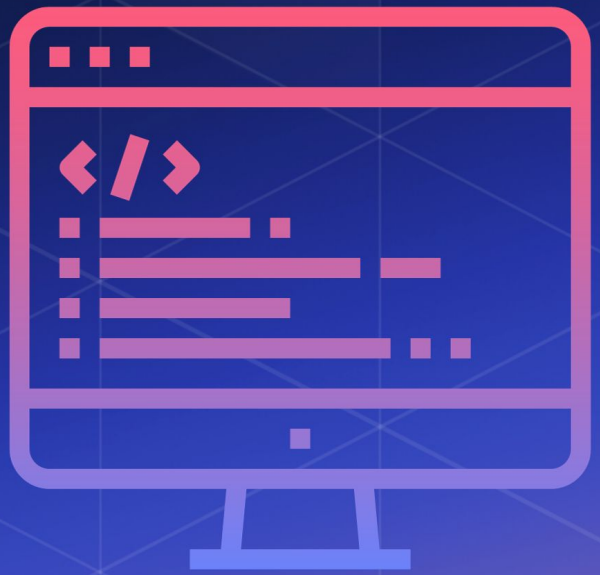
```
$ pip3 install cloudinary-cli
```

Make sure you have the most updated package.

```
$ pip3 install cloudinary-cli --upgrade
```

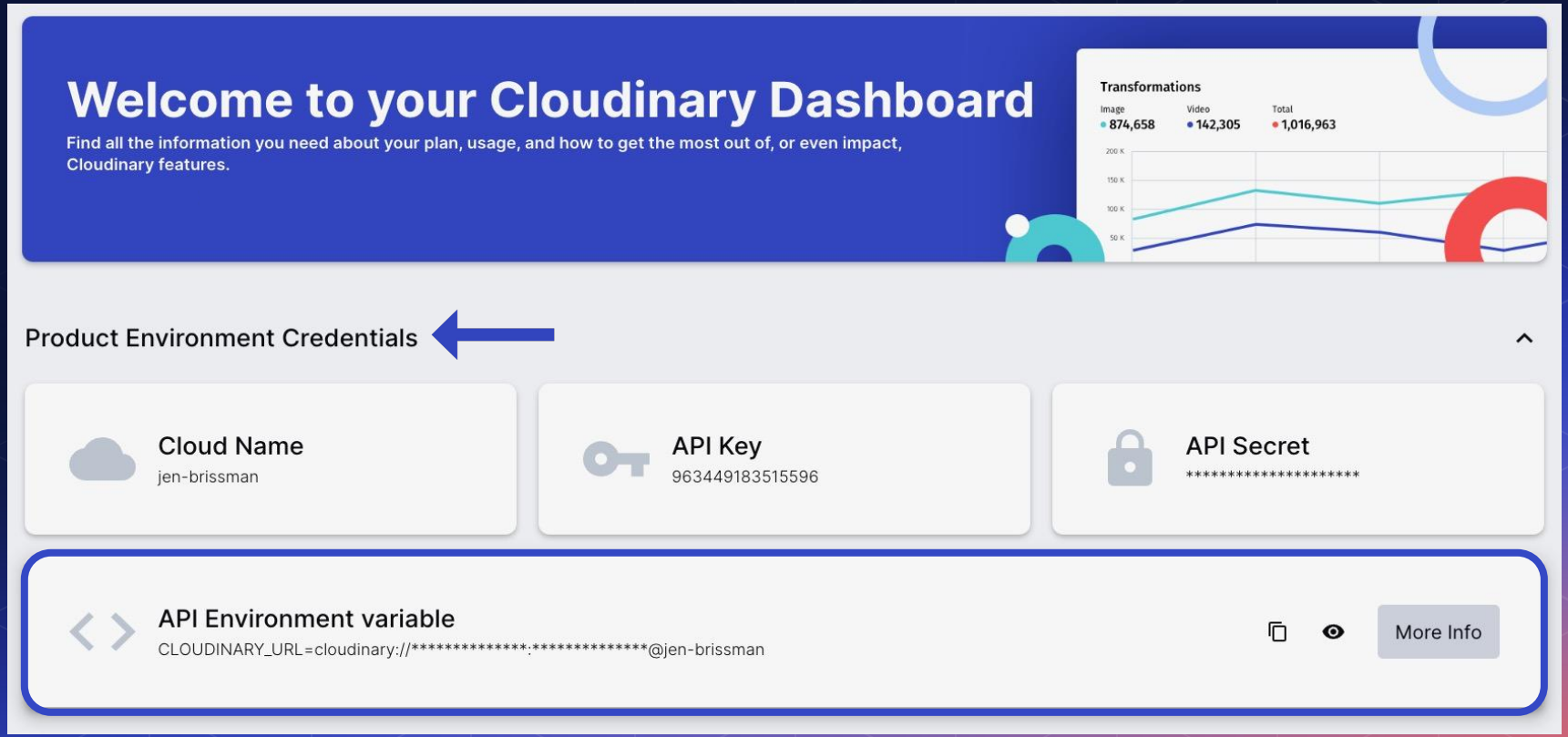


Cloudinary Credentials



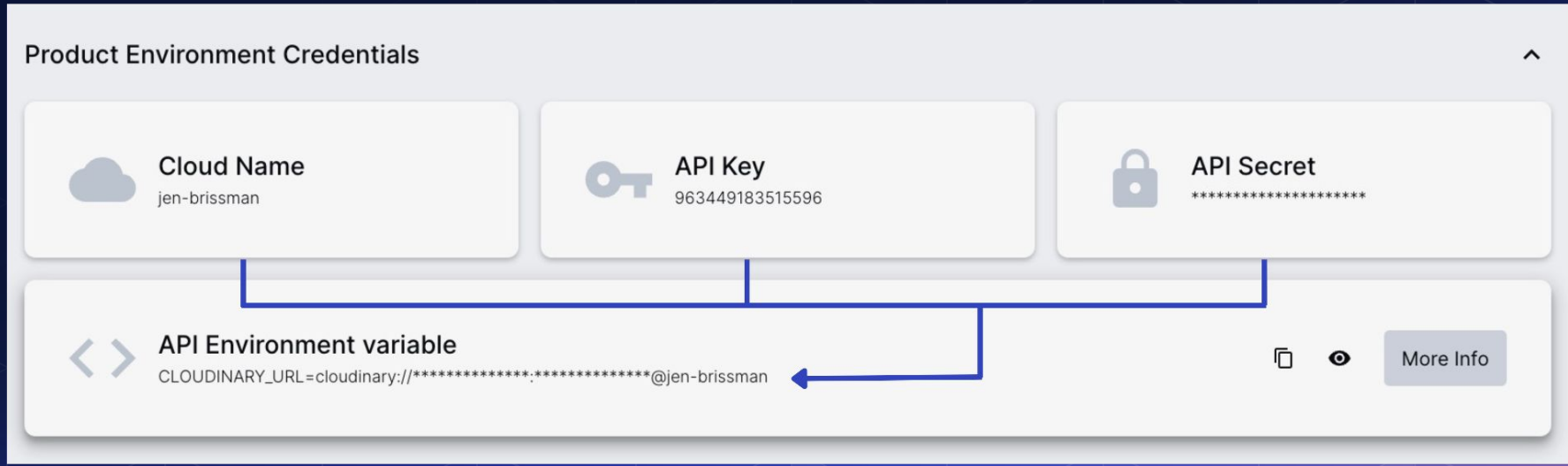
Cloudinary_URL (API Environment Variable)

You can locate your CLOUDINARY_URL on your Cloudinary Dashboard under Product Environment Credentials.



Cloudinary_URL (API Environment Variable)

The CLOUDINARY_URL contains your cloud name, API Key, and API Secret.



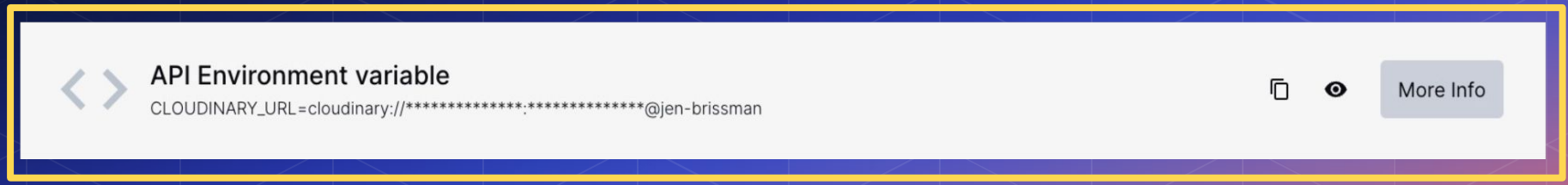
Important!
Do not expose your API Secret in your client-side code, as it creates a security risk and potentially allows others to access your account.

API Environment Variable Configuration

Then **Export**(Mac) or **Set** (Windows) your CLOUDINARY_URL via the CLI.

```
export CLOUDINARY_URL=cloudinary://API_KEY:API_SECRET@cloud_name
```

```
set CLOUDINARY_URL=cloudinary://API_KEY:API_SECRET@cloud_name
```



Cloudinary

Check Cloudinary Configuration

Check your Cloudinary configuration in the CLI

```
$ cld config
cloud_name:    cloud_name
api_key:      API_KEY
api_secret: **************CRET
private_cdn: False
```

Cloudinary

from_url Configuration

You can also configure from the CLI using cld config --from_url, which will save your configuration for use beyond a single session.

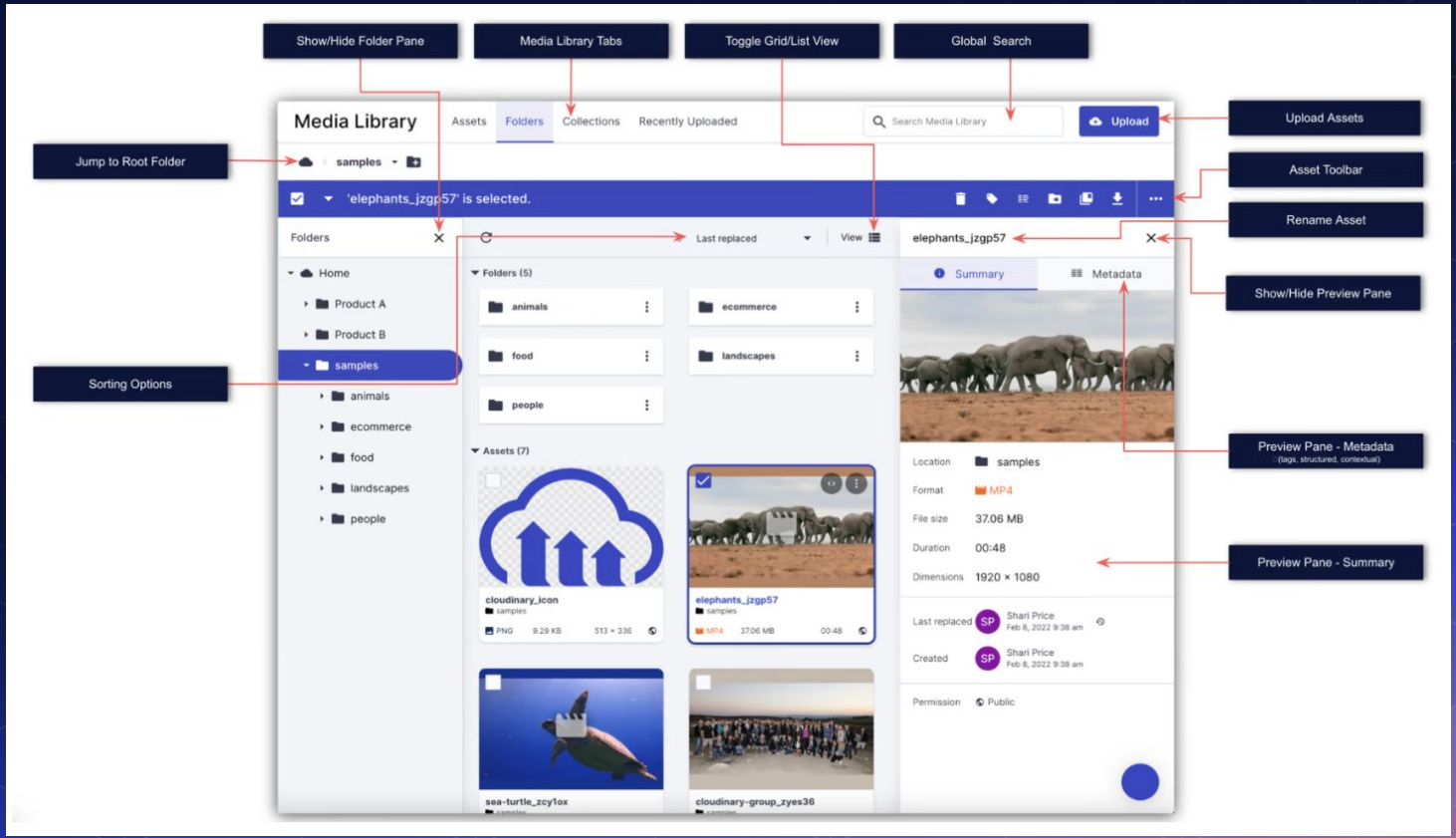
However, future commands would need to be prefixed with cld -C cloudname, instead of cld.

```
$ cld config --from_url cloudinary://API_KEY:API_SECRET@cloud_name

Config cloud_name saved!


$ cld -C cloud_name config

cloud_name:   cloud_name

api_key:   API_KEY

api_secret:      **************CRET
```

You can use cld config --from_url to set up **multiple** account configurations

Cloudinary

Digital Asset Management (DAM)



Cloudinary

Helper Tools

Cloudinary

Helper Tools

If you are looking for help while using the CLI, you can use the following command:

```
$ cld –-help   # see list of commands (below)
```

```
Usage: cld [OPTIONS] COMMAND [ARGS]...

Options:
 --help Show this message and exit.
 --version Show the version and exit.
 -c, --config TEXT Tell the CLI which account to run the command on by specifying an account environment variable.
 -C, --config_saved TEXT Tell the CLI which account to run the command on by specifying a saved configuration
 -v, --verbosity LVL Either CRITICAL, ERROR, WARNING, INFO or DEBUG

Commands:
  admin Run any methods that can be called through the admin API.
  config Display the current configuration, and manage additional...
  make  Return template code for implementing the specified Cloudinary widget.
  migrate Migrate files using an existing auto-upload mapping and a file of URLs.
  provisioning Run any methods that can be called through the provisioning API.
  regen_derived Regenerate all derived assets pertaining to a named transformation, or transformation string.
  search Run the admin API search method.
  sync Synchronize between a local directory and a Cloudinary folder.
  upload_dir Upload a folder of assets, maintaining the folder structure.
  uploader Run any methods that can be called through the upload API.
  url Generate a Cloudinary URL, which you can optionally open...
  utils Call Cloudinary utility methods.
```

Basic Commands

To help you get started, here are some basic commands you can use.

```
$ cld –-version   # see versions of Cloudinary CLI, underlying Cloudinary Python SDK and Python
```

```
$ cld admin   # see available Admin API methods
```

```
$ cld uploader   # see available Upload API methods
```

```
$ cld search –-help   # see usage for the Search API
```

```
$ cld utils   # see available utility methods
```

```
$ cld admin --doc   # opens the Admin API reference in a browser
```

```
$ cld upload --doc   # opens the Upload API reference in a browser
```

Uploading

Uploading

- Upload Method
- Fetch
- Public ID
- Resource Type
- Folders
- JSON Response
- Upload Widget

Cloudinary

Upload Method

```
$ cld upload
```

The *upload method* performs an authenticated upload API call over HTTPS.

Try performing a basic upload of your own from a local folder.

- Navigate within the CLI to the local folder that contains the asset you would like to upload.
- Enter the command cld and the name of the asset you would like to upload.

```
$ cld upload hiking.jpg
```

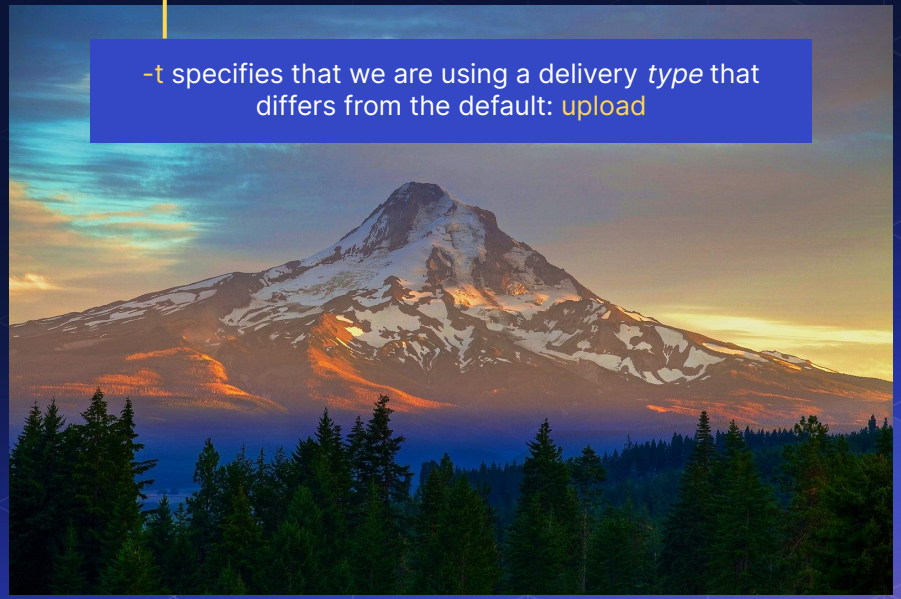
*I am using a file called hiking.jpg

Fetch

You can fetch images from remote sources and upload them straight to Cloudinary entering the command url -t fetch and the URL of the image.

```
cld url -t fetch 'https://upload.wikimedia.org/wikipedia/commons/d/d5/Mt._Hood_%288081466807%29.jpg'
```

-t specifies that we are using a delivery *type* that differs from the default: upload



- 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

Public ID

A unique identifier for your asset that appears in the URL, which is used to reference the uploaded resource as well as for building dynamic delivery and transformation URLs.

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

If you **DO** supply a Public ID, **DO NOT** include the file extension.

$ cld upload mountains.jpg	**Public ID:** “vr1kukfernlipukpw3hr”
$ cld upload mountains.jpg use_filename=true unique_filename=false	**Public ID:** “mountains”
$ cld upload mountains.jpg use_filename=true unique_filename=true	**Public ID:** “mountains_yqnfod”
$ cld upload mountains.jpg public_id=i-love-hiking	**Public ID:** “i-love-hiking”

Public ID in the URL

https://res.cloudinary.com/demo/image/upload/mountains.jpg

Cloudinary

Resource Type

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

You can let Cloudinary determine the type by setting the parameter to “auto”.

```
$ cld upload water-bottle.mp4 resource_type=auto
```

Or you can manually define for each file.

```
$ cld upload water-bottle.mp4 resource_type=video
```

Resource Type in the URL

```
https://res.cloudinary.com/demo/video/upload/water-bottle.mp4
```

Folders

Simply put, folders are a way to organize and divide your files within Cloudinary.

- Folder paths can be included in the **Public ID** parameter

```
$ cld upload dolomites.jpg public_id=italy/dolomites
```

Public ID: “italy/dolomites”

*TIP: A folder will **automatically** be created for you with this call, if it did not already exist.

- Folder paths can also be created in a separate **Folder** parameter

```
$ cld upload dolomites.jpg folder=italy public_id=dolomites
```

Public ID: “italy/dolomites”

- This works in the same way for sub folders

```
$ cld upload dolomites.jpg use_filename=true unique_filename=false folder=mountains/italy
```

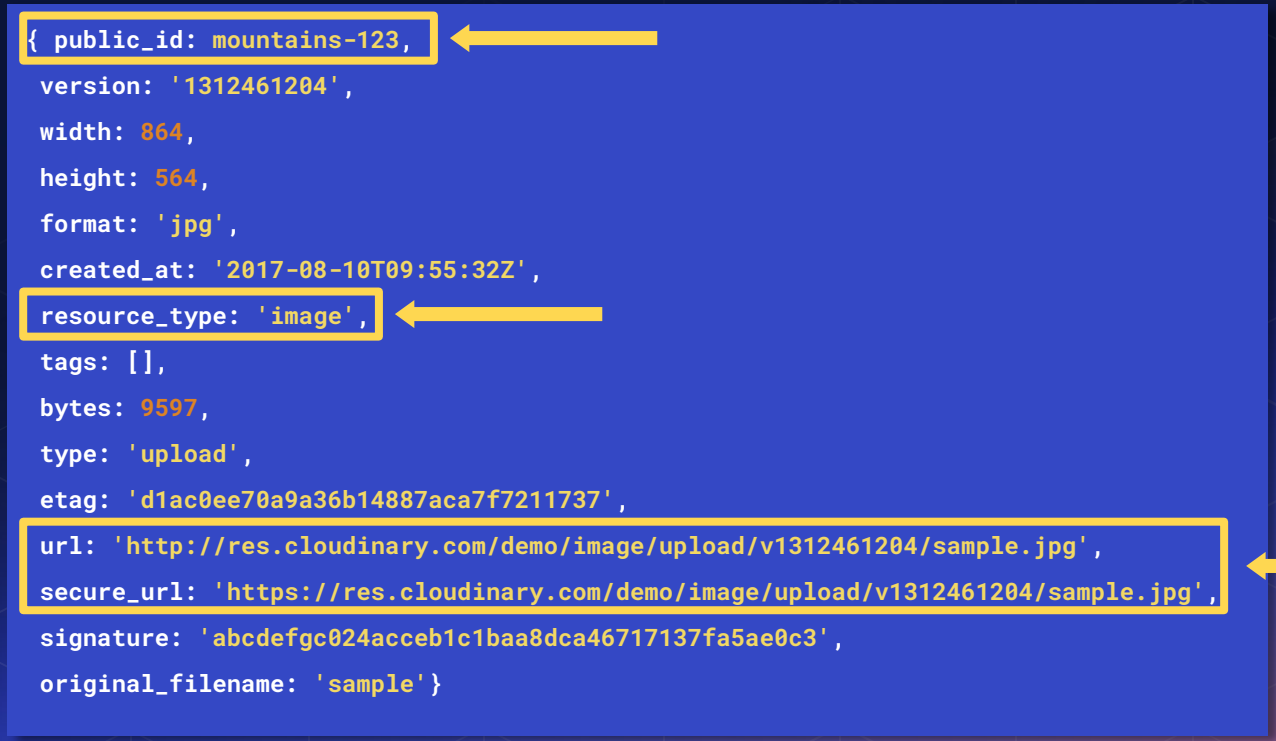
Public ID: “mountains/italy/dolomites”

Folders in the URL

https://res.cloudinary.com/jen-brissman/image/upload/mountains/italy/dolomites.jpg

JSON Response

Cloudinary returns a JSON response that includes HTTP and HTTPS URLs for accessing the uploaded image, as well as additional image information.



Cloudinary

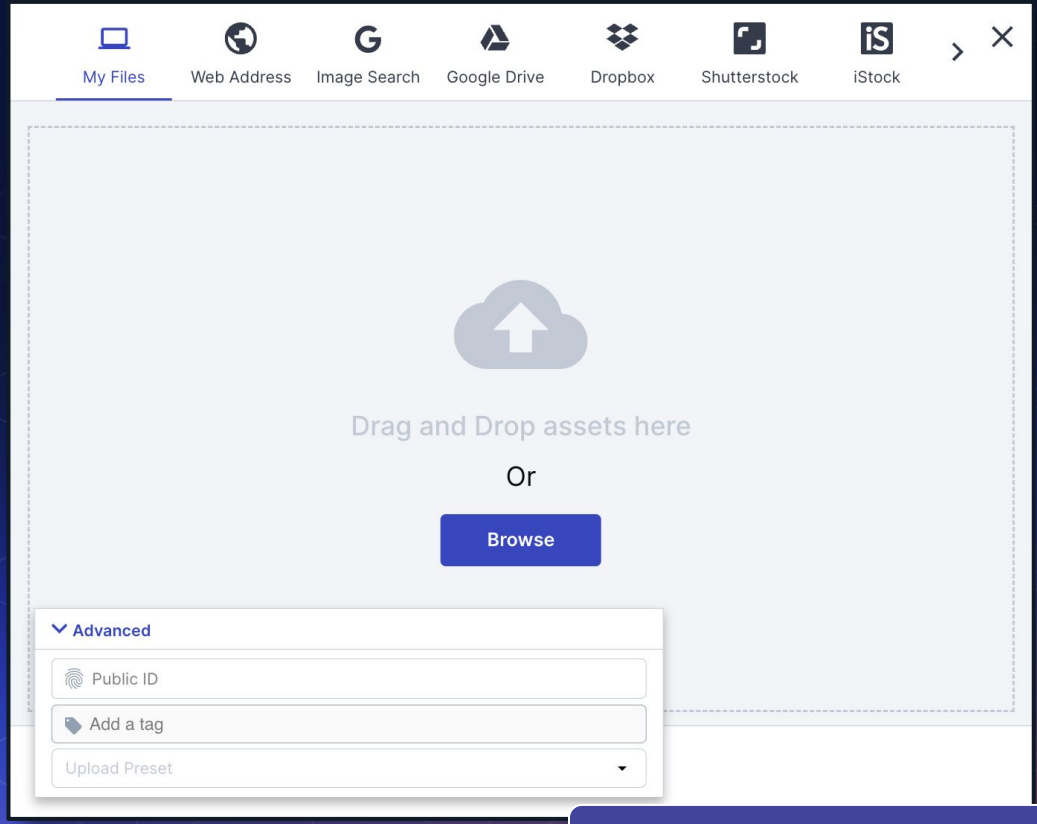
Upload Widget

You can implement Cloudinary's Upload Widget to easily upload assets from the frontend of a web/mobile browser directly to your Cloudinary account, without involving any servers in the process.

The widget shows upload progress and offers thumbnail-preview, as well as interactive-cropping capabilities.

Create an Upload Widget HTML using the CLI

```
$ cld make upload widget
```



Upload Widget Tutorial

Cloudinary URLs

Cloudinary

Cloudinary Delivery URL Structure

Domain	https://res.cloudinary.com/
Cloud Name	<cloud_name>/
Resource Type	<resource_type>/
Delivery Type	<type>/
Transformations	<transformations>/
Version	<version>/
Public ID and Format	<public_id>.<format>

https://res.cloudinary.com/jen-brissman/image/upload/v1702185884/mountains.jpg

<domain>/<cloud_name>/<resource_type>/<type>/<version>/<public_id>.<format>

Cloudinary

Creating Cloudinary URLs

There are two main ways to generate a Cloudinary delivery URL in the CLI. Let's create a URL with the public ID of "mountains" and scale it to a width of 300 pixels.

The Helper command, url, generates a Cloudinary URL.
Note: Unless the URL is opened, the derived asset is not generated.

```
$ cld url mountains w_300,c_scale
```

The utils command enables you to call Cloudinary utility methods. One of these is the cloudinary_url method, for embedding images in web pages using SDKs.

```
$ cld utils cloudinary_url mountains width=300 crop=scale
```

Same exact URL output for both:

```
https://res.cloudinary.com/demo/image/upload/c_scale,w_300/mountains
```

Managing Assets

Cloudinary

Upload API

Admin API

Cloudinary

Managing Assets

- List Resources
- Search
- Rename
- Tags
- Delete
- Invalidate and Versioning

List Resources

Using Admin API methods with the Cloudinary CLI, you can list all of your uploaded assets by many different sets of criteria.

- List all images

```
$ cld admin resources
```

- List asset with a given public ID

```
$ cld admin resource mountain-123
```

- List all images with a given tag

```
$ cld admin resources tags=”blurry”
```

- List all images with a given prefix

```
$ cld admin resources type=upload prefix=sample
```

Search

The search command runs the Admin API search method. This method allows you to filter and retrieve information on all the assets in your product environment with the help of query expressions in a Lucene-like query language.

- Search for images with the tag “blurry”

```
$ cld admin search tags=blurry
```

- Lucene-like query

```
$ cld admin search -f tags -n 30 -s public_id asc "resource_type:image AND
tags=bird AND uploaded_at>1d AND bytes>100k"
```

Rename

Renaming immediately and permanently updates the assets in your cloud storage.

```
$ cld upload rename [old public ID] [new public ID]
```

```
$ cld upload rename mountains tetons
```

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

Tags

With tags you can categorize and organize your assets. You can add tags to your assets at any time, whether during upload or after the asset is in your account.

```
$ cld upload tent.jpg public_id=tent tags="tent, stars, grass"
```

```
$ cld add_tag italy public_ids=tent
```

You can also remove tags to uploaded assets

```
$ cld remove_tag grass public_ids=tent
```

```
$ cld remove_all_tags tent
```

You can also use our AI auto tagging add-ons to further automate the tagging process

```
$ cld uploader upload "tent.jpg" categorization=aws_rek_tagging,google_tagging auto_tagging=".95"
```

Delete

The destroy method of the Upload API immediately destroys and permanently updates the assets in your cloud storage.

```
$ cld uploader destroy tent
```

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

```
$ cld admin delete_resources tetons,i-love-hiking
```

```
$ cld admin delete_resources_by_prefix mountains
```

Invalidate and Versioning

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.
- The old cached media asset **can remain** on the CDN servers for up to 30 days.
- 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.

```
$ cld destroy sample invalidate=true
```

Note: From within the UI/DAM, an invalidate request is **automatically** included whenever you delete, rename, or overwrite media assets.

To check versioning:

```
$ cld admin resource sample versions=true
```

Sync
(Local File Management)

Cloudinary

Sync (Local File Management)

Syncing Local Files and Folders to Cloudinary. Upload from a local directory to the cloud or download a cloud directory to your local file system. This is unique to the CLI, it doesn't exist in SDKs.

Similar to **git** syntax, sync allows you to **push** to the cloud and **pull** from the cloud.

Keep in mind that in this case sync does not mean constantly syncing or “synchronizing”.

For perpetual file syncing check out:
Cloudy Desktop from Cloudinary Labs

Sync (Local File Management)

Sync example - If you have a local directory under your root, “~/Pictures/images” that you want to upload to a folder in your Cloudinary account named “test-images”, you can use the push option.

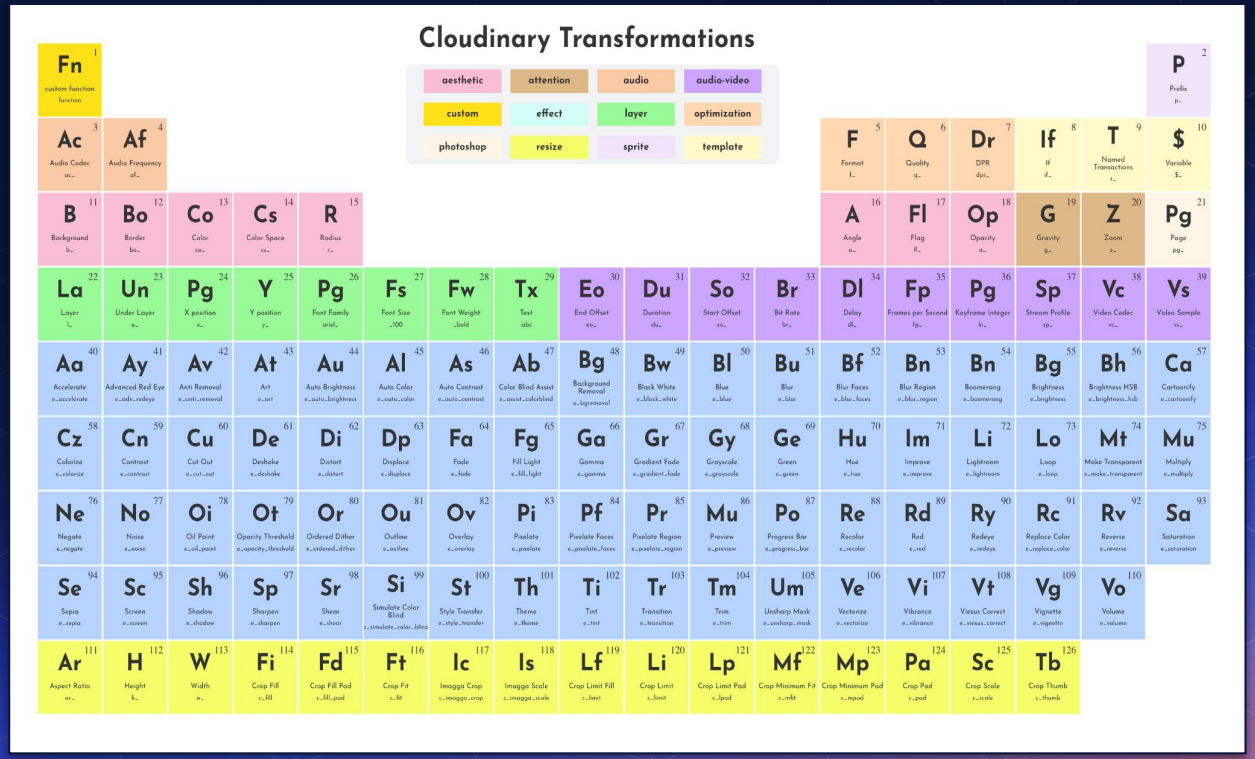
```
$ cld sync --push ~/Desktop/cli-course test-assets
```

If you want to download the “test-assets” folder from Cloudinary to another local directory, for example “~/Desktop/pulled-from-cld”, you can use the pull option.

```
$ cld sync --pull ~/Desktop/pulled-from-cld test-assets
```

Cloudinary
Transformations

Cloudinary Transformations



Cloudinary

Example Assets

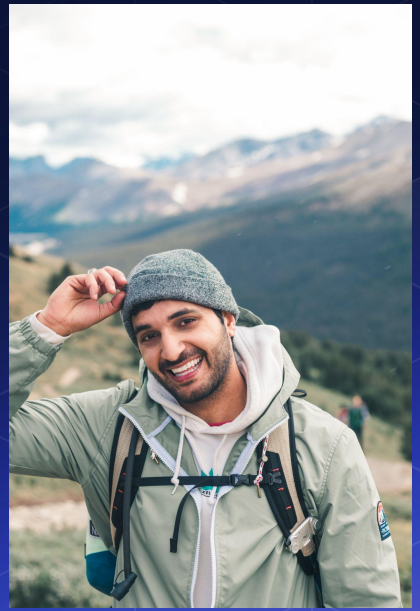
Let’s use this photo to perform various Cloudinary transformations via the CLI.

First, we’ll upload two assets to our account with designated Public IDs of “hiker” and “mountains”

Then we can easily reference the asset by this Public ID throughout this transformations section.

```
$ cld upload hiker.jpg public_id=hiker
```

```
$ cld upload mountains.jpg public_id=mountains
```



Cloudinary

Resize

You can resize any image or video by editing its width (w_) and/or height (h_)

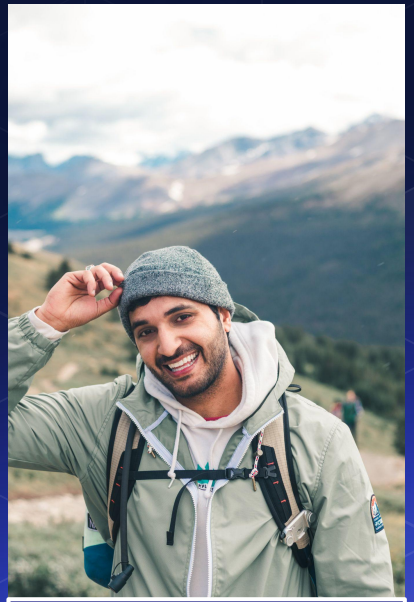
Decreasing the width and/or height of the asset will commonly decrease the file size, optimizing it for a specific project.

```
$ cld url hiker w_300
```

The default is scale. If you don't provide one of the parameters it will automatically scale for you.

You can also make transformations in the URL.

```
http://res.cloudinary.com/jen-brissman/image/upload/w_300/hiker
```



1.05 MB

15.88 KB

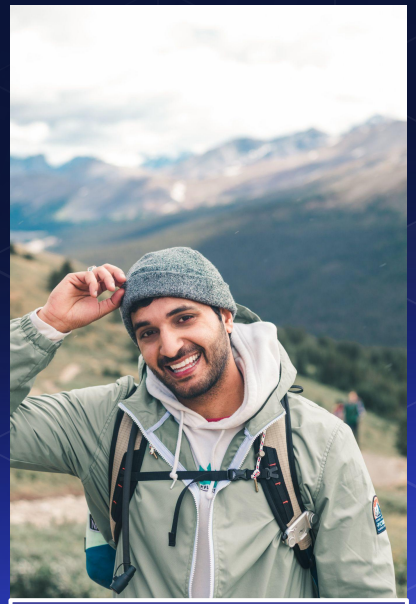
Cloudinary

Format (f_auto)

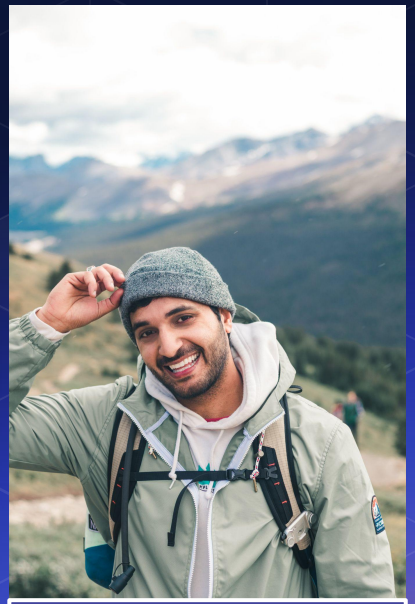
Use Cloudinary to automatically convert assets to other formats for displaying in your web site or application (output formats).

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.



JPG - 1.05 MB



AVIF - 492.4 KB

```
$ cld url hiker f_auto
```

```
http://res.cloudinary.com/jen-brissman/image/upload/f_auto/hiker
```

Cloudinary

Crop with Gravity

You can use Gravity to specify a location in an image or video that is used as the focus for another transformation.

Let's make a 1:1 aspect ratio by adjusting the height and width to be 200 each. Then perform a thumb crop.

We can utilize g_auto to make sure the face is centered in the frame.

```
$ cld url hiker w_500,h_500,c_thumb,g_auto
```

```
https://res.cloudinary.com/jen-brissman/image/upload/w_500,h_500,c_thumb,g_auto,hiker
```

Cloudinary

Rounding for Aesthetics

For rounding, you can use the radius parameter to control the degree of rounding applied to the image.

Here we will use r_max to create a complete circle.

We will also use the grayscale effect, e_grayscale.

Rounded images are commonly used for circular profile pictures, icons, or buttons in user interfaces.

If you have a design theme that involves rounded elements, rounding images can help maintain a consistent visual style throughout your application or website.

```
$ cld url hiker w_500,h_500,c_thumb,g_auto,r_max,e_grayscale,f_auto
```

```
https://res.cloudinary.com/jen-brissman/image/upload/w_500,h_500,c_thumb,g_auto,r_max,e_grayscale,f_auto/hiker
```

Quality (q_auto)

Quality controls the visual quality and compression level of assets.

With q_auto, you allow Cloudinary to deliver the optimal quality of images and videos for each viewing device.

It compresses your asset without a visual difference, but a big difference in the file size as you can see.

$ cld url mountains q_auto

http://res.cloudinary.com/jen-brissman/image/upload/q_auto/mountains

No visual difference

8.53 MB - JPG

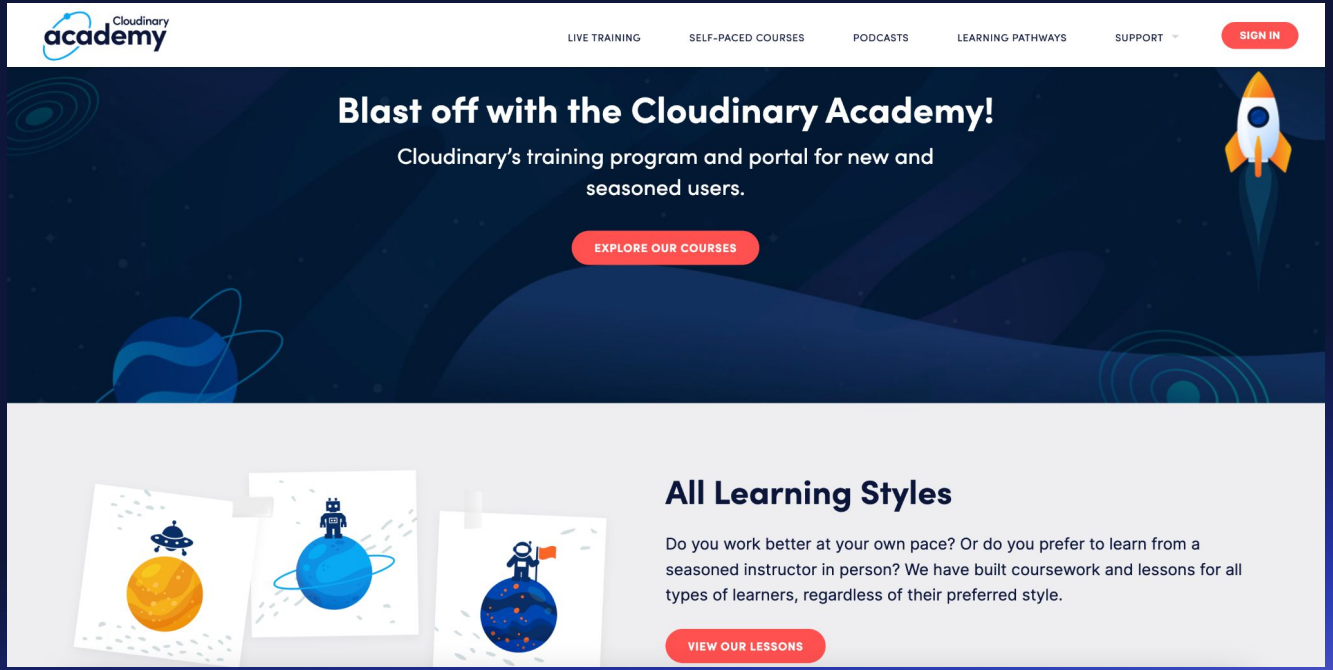
3.88 MB - JPG

Cloudinary

Next Steps and Further Support

Cloudinary Academy

The Cloudinary Academy offers both self-paced courses and live classes taught by Cloudinary experts.



training.cloudinary.com

Cloudinary

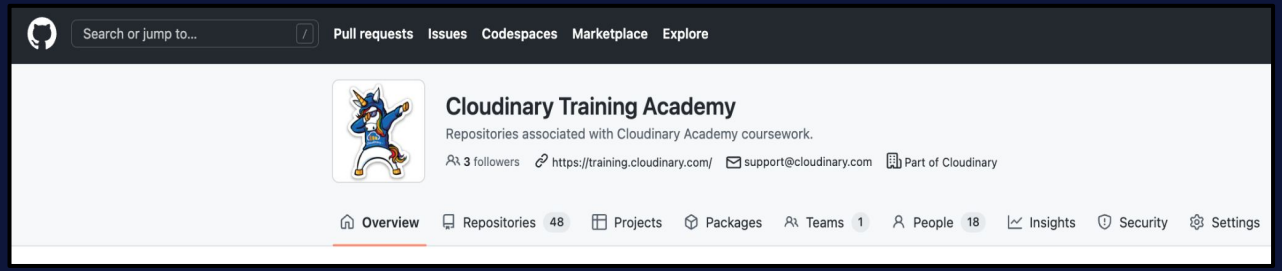
Recommended Courses

Those who are new to Cloudinary's APIs can benefit from a variety of helpful, self-paced courses that provide comprehensive learning resources.

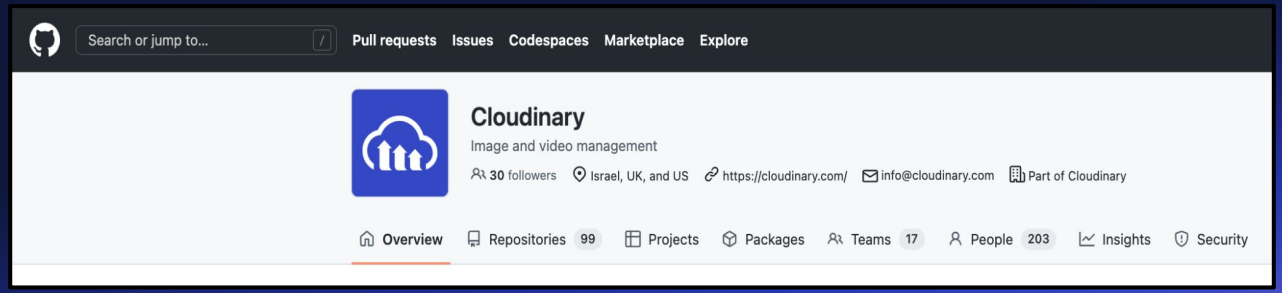
- Introduction to Cloudinary Programmable Media (90-Minutes)
- Understanding Cloudinary Programmable Media Terminology (30-Minutes)
- Cloudinary JumpStart for New Developer Users (~40-Minutes)
- Advanced Concepts for Developers (~16 hours)
- Introduction for Node.js Developers (90 minutes)
- Fundamentals for Developers (~9 hours)

Review our GitHub Repositories

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



github.com/cloudinary-training

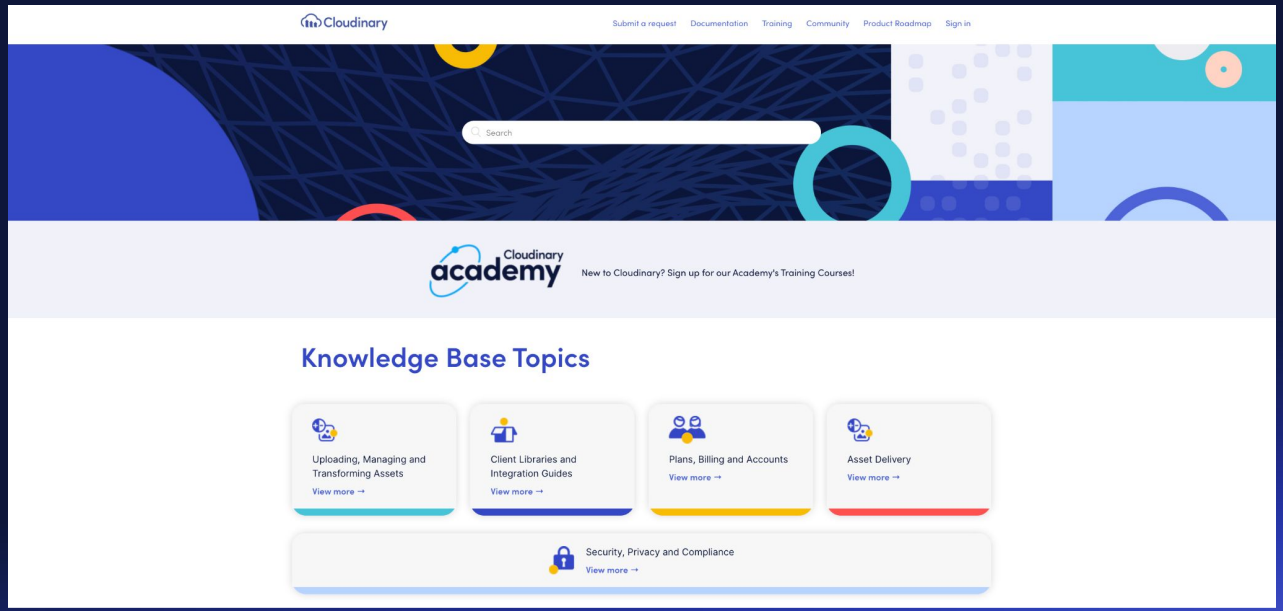


github.com/cloudinary

Cloudinary

Engineering Support

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

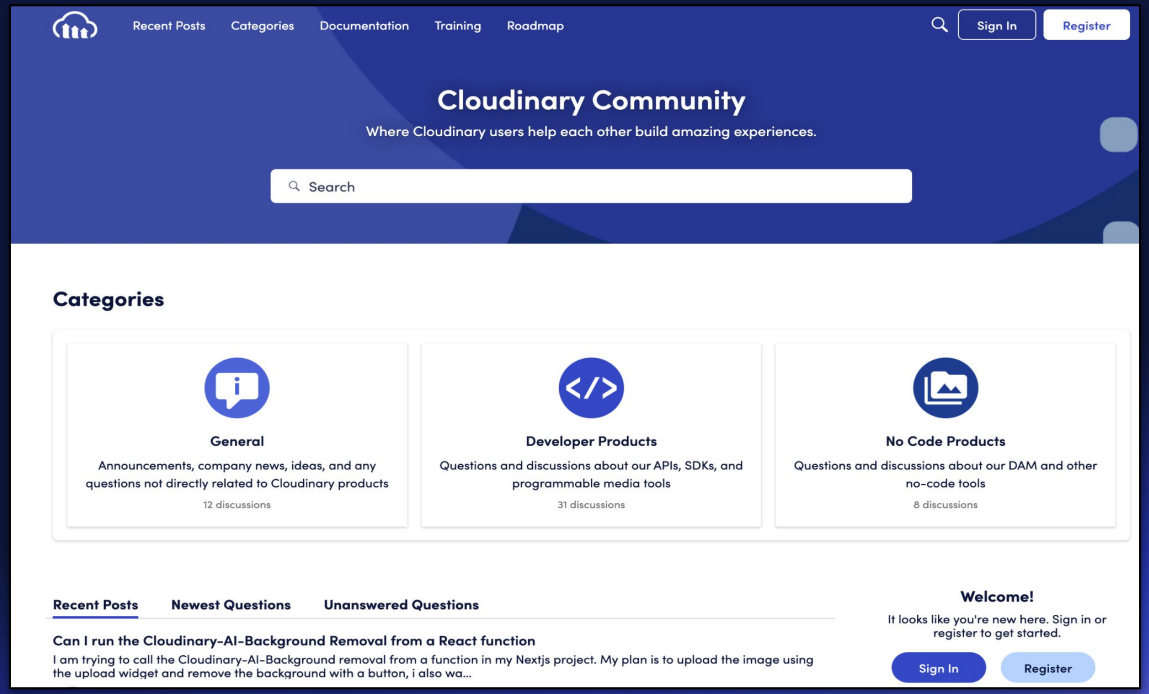


support.cloudinary.com/hc/en-us/requests/new

Cloudinary

Join Our Community Forums

Ask questions to staff or other users in our dedicated communities.

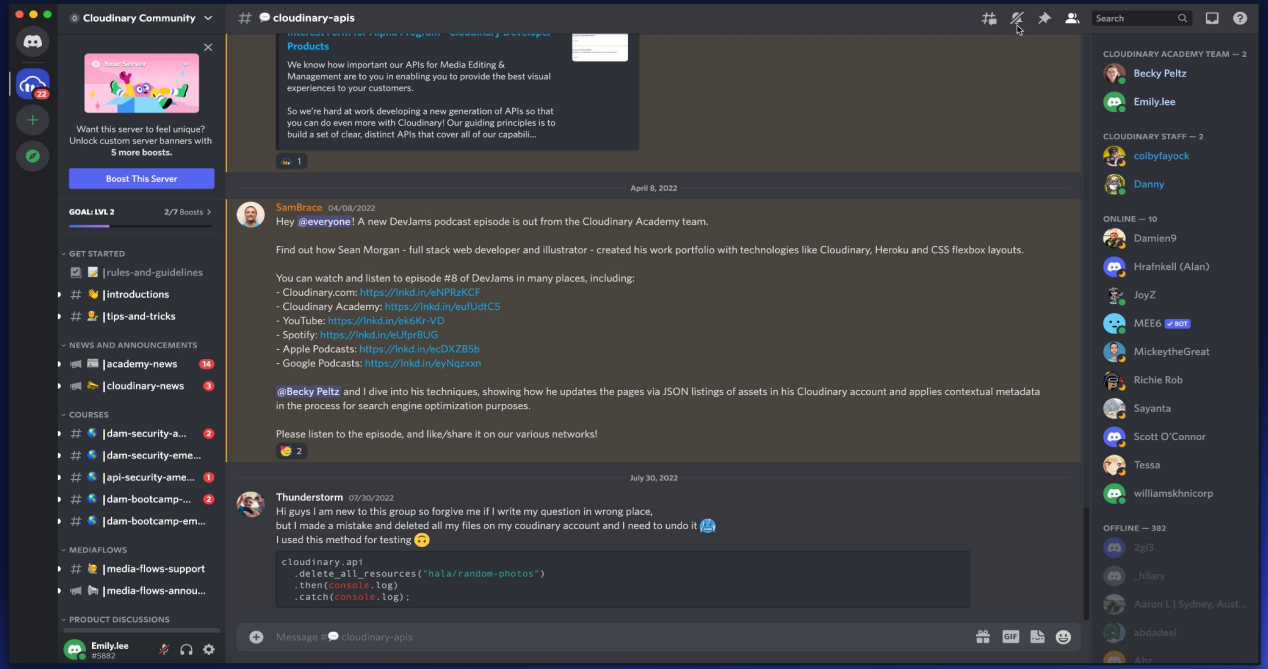


community.cloudinary.com

Cloudinary

Join Our Discord Discussions

Connect with Cloudinary users and staff on Discord!



discord.gg/cloudinary

Cloudinary

Thank you

Cloudinary