Documentation (/documentation) › Integrations (/documentation/framework_integration) › SDKs (/documentation/cloudinary_sdks) › **Node.js SDK**

Node.js SDK

Overview

Cloudinary is a cloud-based service that provides an end-to-end image and video management solution. The Node.js SDK (https://github.com/cloudinary/cloudinary_npm) provides simple, yet comprehensive image and video manipulation, optimization, and delivery capabilities that you can implement using code that integrates seamlessly with your existing Node.js SDK application.

On this page:

Quick example: Transformations

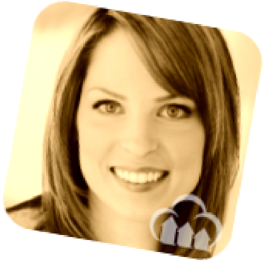
The following Cloudinary URL and corresponding Node.js SDK code delivers the image below in an `https` delivery URL, including all of the following transformations:

- Thumbnail crop to a size of 150x150 pixels using face detection gravity to automatically determine the location for the crop
- Rounded corners with a 20 pixel radius
- Sepia effect
- Overlay of the Cloudinary logo on the southeast corner (with a slight offset). The logo is scaled down to a 50 pixel width, with increased brightness, and partial transparency (opacity = 60%)

- Rotated by 10 degrees
- Converted to and delivered in PNG format (the originally uploaded image was a JPG)

URL Node.js

```
https://res.cloudinary.com/demo/image/upload/w_150,h_150,c_thumb,g_face,r_20,e_sepia/l_cloudinary_icon,g_south_east,x_5,y_5,w_50,o_60,e_brightness:200/a_10/front_
face.png
(https://res.cloudinary.com/demo/image/upload/w_150,h_150,c_thumb,g_face,r_20,e_sepia/l_cloudinary_icon,g_south_east,x_5,y_5,w_50,o_60,e_brightness:200/a_10/front
_face.png)
```



(https://res.cloudinary.com/demo/image/upload/w_150,h_150,c_thumb,g_face,r_20,e_sepia/l_cloudinary_icon,g_south_east,x_5,y_5,w_50,o_60,e_brightness:200/a_10/front_face.png)

Quick example: File upload

The following Node.js code uploads the `dog.mp4` video to the specified account sub-folder using the public_id, `my_dog` . The video will overwrite the existing `my_dog` video if it exists. When the video upload is complete, the specified notification URL will receive details about the uploaded media asset.

```
cloudinary.v2.uploader.upload("dog.mp4",
  {resource_type: "video", public_id: "my_folder/my_sub_folder/my_dog",
  overwrite: true, notification_url: "https://mysite.example.com/notify_endpoint"},
  function(error, result) {console.log(result, error)});
```

Node.js SDK features

- Build URLs for image and video manipulations
- Node.js helper methods for embedding and transforming images
- API wrappers: file upload, administration, sprite generation and more
- Server-side file upload + direct unsigned file upload from the browser using the jQuery plugin
- TypeScript support. View the TypeScript declaration file (https://github.com/cloudinary/cloudinary_npm/blob/master/types/index.d.ts).

Node.js capitalization and data type guidelines

When using the Node.js SDK, keep these guidelines in mind:

- Parameter names: `snake_case` . For example: **public_id**
- Classes: `PascalCase` . For example: **PreloadedFile**
- Methods: `snake_case` . For example: **image_upload_tag**
- Pass parameter data as: `Object`

Installation and setup

Cloudinary's Node.js integration library is available as an open-source NPM. To install the Cloudinary NPM, run:

```
npm install cloudinary
```

Include Cloudinary's Node.js classes in your code:

```
var cloudinary = require('cloudinary');
```

Important: The Node.js SDK upload and admin method syntax examples shown throughout this documentation use the **v2** signature. To avoid confusion, all code examples are shown in the format `cloudinary.v2.:...` .

In your own code, it is recommended to include `v2` of the Node.js classes as follows:

```
var cloudinary = require('cloudinary').v2;
```

Following this, your upload and admin API calls should omit the `.v2` shown in the code examples of this guide.
For example, a simple image upload:

```
cloudinary.uploader.upload("my_image.jpg", function(error, result) {console.log(result, error)});
```

Configuration

To use the Cloudinary Node.js SDK, you have to configure at least your `cloud_name`. Your `api_key` and `api_secret` are also needed for secure API calls to Cloudinary (e.g., image and video uploads). You can find your account-specific configuration credentials in the **Dashboard** page of the account console (/console).

Setting the configuration parameters can be done either programmatically in each call to a Cloudinary method or you can set them globally using either an environment variable or the `config` method. You can optionally configure the required `cloud_name`, `api_key`, and `api_secret` by defining the **CLOUDINARY_URL** environment variable. The configuration URL is available in the **Dashboard** page of the account console. When using Cloudinary through a PaaS add-on (e.g., Heroku), this environment variable is automatically defined in your deployment environment. For example:

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

In addition to the required configuration parameters, you can define a number of optional configuration parameters (cloudinary_sdks#configuration_parameters) if relevant.

Here's an example of setting configuration parameters globally in your Node application:

```
cloudinary.config({
  cloud_name: 'sample',
  api_key: '874837483274837',
  api_secret: 'a676b67565c6767a6767d6767f676fe1'
});
```

Sample projects

For additional useful code samples and to learn how to integrate Cloudinary with your Node.js applications, take a look at our Sample Projects (https://github.com/cloudinary/cloudinary_npm/tree/master/samples).

- **Basic Node.js sample**: Uploading local and remote images to Cloudinary and generating various transformation URLs.
- **Node.js Photo Album**: A fully working web application that allows you to upload photos, maintain a database with references, list images with their metadata, and display them using various cloud-based transformations. Image uploading is performed both from the server side and directly from the browser using a jQuery plugin.

Related topics

→ Learn more about uploading images and videos (node_image_and_video_upload) using the Node.js SDK.

→ See examples of powerful image (node_image_manipulation) and video (node_video_manipulation) manipulations using Node.js code, and see our image transformations (image_transformations) and video manipulation (video_manipulation_and_delivery) docs.

→ Check out Cloudinary's asset administration (node_asset_administration) capabilities.

→ Stay tuned for updates, tips and tutorials in Product Updates (https://cloudinary.com/product_updates) and Blog Posts (https://cloudinary.com/blog).

← **Django asset administration(/documentation/django_asset_administration)** **Node.js image and video upload(/documentation/node_image_and_video_upload)** →

(https://cloudinary.com/)

Service

Solutions (/solutions)

Pricing (/pricing)

Customers (/customers)

FAQ (/faq)

Resources

Documentation (https://cloudinary.com/documentation)

Blog (https://cloudinary.com/blog)

Webinars (/webinars)

Cookbook (https://cloudinary.com/cookbook)

Visual Web (https://cloudinary.com/visualweb)

Demos (https://cloudinary.com/demos)

Support (https://support.cloudinary.com/hc/en-us)

Service status (https://status.cloudinary.com)

Company

About us (/about)

Team (/team)

Contact us (/contact)

Partners (/partners)

Newsroom (/newsroom)

Careers (/jobs)

Brand assets (/assets)

Trust (/trust)

Connect

(https://www.facebook.com/Cloudinary)

(https://twitter.com/cloudinary)

(https://www.linkedin.com/company/cloudinary/)

Terms of Use (https://cloudinary.com/tos) Privacy Policy (https://cloudinary.com/privacy) DMCA Notice (https://cloudinary.com/dmca)