n8n

-

Cas pratiques
Volume 1

Table des matières

Générer des images disque avec n8n : Convertir des archives en formats FAT et IMG

Explorez comment n8n peut être utilisé pour créer des images disque compatibles FAT ou IMG à partir de fichiers d’archive. Ce guide pratique vous montre comment développer des nœuds personnalisés pour étendre les capacités d’automatisation de n8n.

To create a custom n8n.io node for generating a FAT or FAT32 compatible image from an ISO image, you can follow these steps:
Install the required dependencies:
npm install --save n8n-core n8n-workflow
Create a new file called FatImageFromIso.node.ts and add the following code:

```
import { IExecuteFunctions } from 'n8n-core';
import { INodeExecutionData, INodeType, INodeTypeDescription } from 'n8n-workflow';

export class FatImageFromIso implements INodeType {
  description: INodeTypeDescription = {
    displayName: 'FAT Image from ISO',
    name: 'fatImageFromIso',
    group: ['transform'],
    version: 1,
    description: 'Generate a FAT or FAT32 compatible image from an ISO image',
    defaults: {
      name: 'FAT Image from ISO',
      color: '#772244',
    },
    inputs: ['main'],
    outputs: ['main'],
    properties: [
      {
        displayName: 'ISO Image Path',
        name: 'isoImagePath',
        type: 'string',
        default: '',
        placeholder: '/path/to/your/iso-image.iso',
        description: 'Path to the ISO image file.',
        required: true,
      },
      {
        displayName: 'Output Image Path',
        name: 'outputImagePath',
        type: 'string',
        default: '',
        placeholder: '/path/to/your/output-image.img',
        description: 'Path to save the generated FAT or FAT32 compatible image.',
        required: true,
      },
    ],
  };

  async execute(this: IExecuteFunctions): Promise<INodeExecutionData[][]> {
    const items = this.getInputData();
    const returnData: INodeExecutionData[] = [];

    for (let i = 0; i < items.length; i++) {
      const isoImagePath = this.getNodeParameter('isoImagePath', i) as string;
      const outputImagePath = this.getNodeParameter('outputImagePath', i) as string;

      // Implement the logic to convert the ISO image to a FAT or FAT32 compatible image
      // You may need to use external libraries or tools to achieve this

      returnData.push({
        json: {
```

```
                isoImagePath,
                outputImagePath,
                status: 'success',
            },
        });
    }

    return this.prepareOutputData(returnData);
  }
}
```

Register the new node in n8n.nodes.ts:

```
import { FatImageFromIso } from './FatImageFromIso.node';

export const NodeTypes = [
  // ...
  new FatImageFromIso(),
];
```

Please note that the actual logic to convert the ISO image to a FAT or FAT32 compatible image is not implemented in this code. You may need to use external libraries or tools to achieve this. The code provided here serves as a starting point for creating a custom n8n.io node.

You can use the dd command to create an empty FAT32 image and then mount both the ISO and the FAT32 image to copy the files. Here's a bash script to achieve this:

```
#!/bin/bash

# Check if the user provided the required arguments
if [ "$#" -ne 2 ]; then
  echo "Usage: $0 input.iso output.img"
  exit 1
fi

# Set input and output paths
input_iso="$1"
output_img="$2"

# Calculate the size of the input ISO in 1M blocks
iso_size=$(du -m "$input_iso" | cut -f1)

# Create an empty FAT32 image with the same size as the input ISO
dd if=/dev/zero of="$output_img" bs=1M count="$iso_size"
mkfs.fat -F 32 "$output_img"

# Create temporary mount points for the ISO and the FAT32 image
iso_mount=$(mktemp -d)
img_mount=$(mktemp -d)

# Mount the ISO and the FAT32 image
sudo mount -o loop "$input_iso" "$iso_mount"
sudo mount -o loop "$output_img" "$img_mount"

# Copy the files from the ISO to the FAT32 image
sudo cp -r "$iso_mount"/* "$img_mount"/

# Unmount the ISO and the FAT32 image
sudo umount "$iso_mount"
sudo umount "$img_mount"

# Remove the temporary mount points
rmdir "$iso_mount"
rmdir "$img_mount"

echo "FAT32 image created successfully: $output_img"
```

Save this script as iso_to_fat32.sh and make it executable:

```
chmod +x iso_to_fat32.sh
```

Now you can use the script to convert an ISO image to a FAT32 image:

./iso_to_fat32.sh input.iso output.img
Please note that this script requires sudo privileges to mount and unmount the images. Also, ensure that you have the dosfstools package installed on your system to use the mkfs.fat command. You can install it using:
sudo apt-get install dosfstools
To create a custom n8n.io node for generating an ISO image from a zip archive, you can follow these steps:
Install the required dependencies:
npm install --save n8n-core n8n-workflow
Create a new file called IsoImageFromZip.node.ts and add the following code:

```
import { IExecuteFunctions } from 'n8n-core';
import { INodeExecutionData, INodeType, INodeTypeDescription } from 'n8n-workflow';
import * as fs from 'fs';
import * as path from 'path';
import * as AdmZip from 'adm-zip';
import { exec } from 'child_process';

export class IsoImageFromZip implements INodeType {
  description: INodeTypeDescription = {
    displayName: 'ISO Image from Zip',
    name: 'isoImageFromZip',
    group: ['transform'],
    version: 1,
    description: 'Generate an ISO image from a zip archive',
    defaults: {
      name: 'ISO Image from Zip',
      color: '#772244',
    },
    inputs: ['main'],
    outputs: ['main'],
    properties: [
      {
        displayName: 'Zip Archive Path',
        name: 'zipArchivePath',
        type: 'string',
        default: '',
        placeholder: '/path/to/your/zip-archive.zip',
        description: 'Path to the zip archive file.',
        required: true,
      },
      {
        displayName: 'Output ISO Path',
        name: 'outputIsoPath',
        type: 'string',
        default: '',
        placeholder: '/path/to/your/output-image.iso',
        description: 'Path to save the generated ISO image.',
        required: true,
      },
    ],
  };

  async execute(this: IExecuteFunctions): Promise<INodeExecutionData[][]> {
    const items = this.getInputData();
    const returnData: INodeExecutionData[] = [];

    for (let i = 0; i < items.length; i++) {
      const zipArchivePath = this.getNodeParameter('zipArchivePath', i) as string;
      const outputIsoPath = this.getNodeParameter('outputIsoPath', i) as string;

      // Extract the zip archive to a temporary directory
      const tempDir = fs.mkdtempSync(path.join(__dirname, 'isoImageFromZip-'));
      const zip = new AdmZip(zipArchivePath);
      zip.extractAllTo(tempDir, true);

      // Generate the ISO image using the genisoimage command
```

```
        const genIsoImage = (sourceDir: string, outputPath: string) => {
            return new Promise<void>((resolve, reject) => {
                exec(`genisoimage -o "${outputPath}" -r -J "${sourceDir}"`, (error, stdout, stderr) => {
                    if (error) {
                        reject(error);
                    } else {
                        resolve();
                    }
                });
            });
        };

        try {
            await genIsoImage(tempDir, outputIsoPath);
            returnData.push({
                json: {
                    zipArchivePath,
                    outputIsoPath,
                    status: 'success',
                },
            });
        } catch (error) {
            returnData.push({
                json: {
                    zipArchivePath,
                    outputIsoPath,
                    status: 'error',
                    error: error.message,
                },
            });
        } finally {
            // Clean up the temporary directory
            fs.rmSync(tempDir, { recursive: true, force: true });
        }
    }

    return this.prepareOutputData(returnData);
  }
}
```

Register the new node in n8n.nodes.ts:

```
import { IsoImageFromZip } from './IsoImageFromZip.node';

export const NodeTypes = [
  // ...
  new IsoImageFromZip(),
];
```

Please note that this code requires the genisoimage command-line tool to be installed on your system. You can install it using:

```
sudo apt-get install genisoimage
```

Additionally, you'll need to install the adm-zip package to handle zip archives:

```
npm install --save adm-zip
```

The code provided here serves as a starting point for creating a custom n8n.io node. You may need to adjust the code to fit your specific requirements and environment.

To create a custom n8n.io node for generating an ISO image from a tar archive, you can follow these steps:

Install the required dependencies:

```
npm install --save n8n-core n8n-workflow tar-fs
```

Create a new file called IsoImageFromTar.node.ts and add the following code:

```
import { IExecuteFunctions } from 'n8n-core';
import { INodeExecutionData, INodeType, INodeTypeDescription } from 'n8n-workflow';
import * as fs from 'fs';
import * as path from 'path';
import * as tar from 'tar-fs';
import { exec } from 'child_process';

export class IsoImageFromTar implements INodeType {
```

```
description: INodeTypeDescription = {
    displayName: 'ISO Image from Tar',
    name: 'isoImageFromTar',
    group: ['transform'],
    version: 1,
    description: 'Generate an ISO image from a tar archive',
    defaults: {
        name: 'ISO Image from Tar',
        color: '#772244',
    },
    inputs: ['main'],
    outputs: ['main'],
    properties: [
        {
            displayName: 'Tar Archive Path',
            name: 'tarArchivePath',
            type: 'string',
            default: '',
            placeholder: '/path/to/your/tar-archive.tar',
            description: 'Path to the tar archive file.',
            required: true,
        },
        {
            displayName: 'Output ISO Path',
            name: 'outputIsoPath',
            type: 'string',
            default: '',
            placeholder: '/path/to/your/output-image.iso',
            description: 'Path to save the generated ISO image.',
            required: true,
        },
    ],
};

async execute(this: IExecuteFunctions): Promise<INodeExecutionData[][]> {
    const items = this.getInputData();
    const returnData: INodeExecutionData[] = [];

    for (let i = 0; i < items.length; i++) {
        const tarArchivePath = this.getNodeParameter('tarArchivePath', i) as string;
        const outputIsoPath = this.getNodeParameter('outputIsoPath', i) as string;

        // Extract the tar archive to a temporary directory
        const tempDir = fs.mkdtempSync(path.join(__dirname, 'isoImageFromTar-'));
        const extract = tar.extract(tempDir);
        const tarReadStream = fs.createReadStream(tarArchivePath);
        tarReadStream.pipe(extract);

        await new Promise<void>((resolve, reject) => {
            extract.on('finish', resolve);
            extract.on('error', reject);
        });

        // Generate the ISO image using the genisoimage command
        const genIsoImage = (sourceDir: string, outputPath: string) => {
            return new Promise<void>((resolve, reject) => {
                exec(`genisoimage -o "${outputPath}" -r -J "${sourceDir}"`, (error, stdout, stderr) => {
                    if (error) {
                        reject(error);
                    } else {
                        resolve();
                    }
                });
            });
```

```
        };

        try {
          await genIsoImage(tempDir, outputIsoPath);
          returnData.push({
            json: {
              tarArchivePath,
              outputIsoPath,
              status: 'success',
            },
          });
        } catch (error) {
          returnData.push({
            json: {
              tarArchivePath,
              outputIsoPath,
              status: 'error',
              error: error.message,
            },
          });
        } finally {
          // Clean up the temporary directory
          fs.rmSync(tempDir, { recursive: true, force: true });
        }
      }

      return this.prepareOutputData(returnData);
    }
}
```

Register the new node in n8n.nodes.ts:

```
import { IsoImageFromTar } from './IsoImageFromTar.node';

export const NodeTypes = [
  // ...
  new IsoImageFromTar(),
];
```

Please note that this code requires the genisoimage command-line tool to be installed on your system. You can install it using:

```
sudo apt-get install genisoimage
```

The code provided here serves as a starting point for creating a custom n8n.io node. You may need to adjust the code to fit your specific requirements and environment.

To create a custom n8n.io node for generating an IMG disk image from a zip archive, you can follow these steps:

Install the required dependencies:

```
npm install --save n8n-core n8n-workflow adm-zip
```

Create a new file called ImgImageFromZip.node.ts and add the following code:

```
import { IExecuteFunctions } from 'n8n-core';
import { INodeExecutionData, INodeType, INodeTypeDescription } from 'n8n-workflow';
import * as fs from 'fs';
import * as path from 'path';
import * as AdmZip from 'adm-zip';
import { exec } from 'child_process';

export class ImgImageFromZip implements INodeType {
  description: INodeTypeDescription = {
    displayName: 'IMG Image from Zip',
    name: 'imgImageFromZip',
    group: ['transform'],
    version: 1,
    description: 'Generate an IMG disk image from a zip archive',
    defaults: {
      name: 'IMG Image from Zip',
      color: '#772244',
    },
    inputs: ['main'],
    outputs: ['main'],
```

```
    properties: [
        {
            displayName: 'Zip Archive Path',
            name: 'zipArchivePath',
            type: 'string',
            default: '',
            placeholder: '/path/to/your/zip-archive.zip',
            description: 'Path to the zip archive file.',
            required: true,
        },
        {
            displayName: 'Output IMG Path',
            name: 'outputImgPath',
            type: 'string',
            default: '',
            placeholder: '/path/to/your/output-image.img',
            description: 'Path to save the generated IMG disk image.',
            required: true,
        },
    ],
};

async execute(this: IExecuteFunctions): Promise<INodeExecutionData[][]> {
    const items = this.getInputData();
    const returnData: INodeExecutionData[] = [];

    for (let i = 0; i < items.length; i++) {
        const zipArchivePath = this.getNodeParameter('zipArchivePath', i) as string;
        const outputImgPath = this.getNodeParameter('outputImgPath', i) as string;

        // Extract the zip archive to a temporary directory
        const tempDir = fs.mkdtempSync(path.join(__dirname, 'imgImageFromZip-'));
        const zip = new AdmZip(zipArchivePath);
        zip.extractAllTo(tempDir, true);

        // Generate the IMG disk image using the dd command
        const createImgImage = (sourceDir: string, outputPath: string) => {
            return new Promise<void>((resolve, reject) => {
                exec(`dd if="${sourceDir}" of="${outputPath}"`, (error, stdout, stderr) => {
                    if (error) {
                        reject(error);
                    } else {
                        resolve();
                    }
                });
            });
        };

        try {
            await createImgImage(tempDir, outputImgPath);
            returnData.push({
                json: {
                    zipArchivePath,
                    outputImgPath,
                    status: 'success',
                },
            });
        } catch (error) {
            returnData.push({
                json: {
                    zipArchivePath,
                    outputImgPath,
                    status: 'error',
                    error: error.message,
```

```
                },
            });
        } finally {
            // Clean up the temporary directory
            fs.rmSync(tempDir, { recursive: true, force: true });
        }
    }

    return this.prepareOutputData(returnData);
  }
}
```

Register the new node in n8n.nodes.ts:

```
import { ImgImageFromZip } from './ImgImageFromZip.node';

export const NodeTypes = [
  // ...
  new ImgImageFromZip(),
];
```

Please note that the code provided here serves as a starting point for creating a custom n8n.io node. The actual logic to convert the extracted zip archive to an IMG disk image might need to be adjusted depending on the specific requirements and environment. The current implementation uses the dd command, which might not be suitable for all use cases. You may need to use external libraries or tools to achieve the desired result.

To create a custom n8n.io node for generating an IMG disk image from a tar archive, you can follow these steps:

Install the required dependencies:

```
npm install --save n8n-core n8n-workflow tar-fs
```

Create a new file called ImgImageFromTar.node.ts and add the following code:

```
import { IExecuteFunctions } from 'n8n-core';
import { INodeExecutionData, INodeType, INodeTypeDescription } from 'n8n-workflow';
import * as fs from 'fs';
import * as path from 'path';
import * as tar from 'tar-fs';
import { exec } from 'child_process';

export class ImgImageFromTar implements INodeType {
  description: INodeTypeDescription = {
    displayName: 'IMG Image from Tar',
    name: 'imgImageFromTar',
    group: ['transform'],
    version: 1,
    description: 'Generate an IMG disk image from a tar archive',
    defaults: {
      name: 'IMG Image from Tar',
      color: '#772244',
    },
    inputs: ['main'],
    outputs: ['main'],
    properties: [
      {
        displayName: 'Tar Archive Path',
        name: 'tarArchivePath',
        type: 'string',
        default: '',
        placeholder: '/path/to/your/tar-archive.tar',
        description: 'Path to the tar archive file.',
        required: true,
      },
      {
        displayName: 'Output IMG Path',
        name: 'outputImgPath',
        type: 'string',
        default: '',
        placeholder: '/path/to/your/output-image.img',
        description: 'Path to save the generated IMG disk image.',
        required: true,
```

```
        },
      ],
    };

    async execute(this: IExecuteFunctions): Promise<INodeExecutionData[][]> {
      const items = this.getInputData();
      const returnData: INodeExecutionData[] = [];

      for (let i = 0; i < items.length; i++) {
        const tarArchivePath = this.getNodeParameter('tarArchivePath', i) as string;
        const outputImgPath = this.getNodeParameter('outputImgPath', i) as string;

        // Extract the tar archive to a temporary directory
        const tempDir = fs.mkdtempSync(path.join(__dirname, 'imgImageFromTar-'));
        const extract = tar.extract(tempDir);
        const tarReadStream = fs.createReadStream(tarArchivePath);
        tarReadStream.pipe(extract);

        await new Promise<void>((resolve, reject) => {
          extract.on('finish', resolve);
          extract.on('error', reject);
        });

        // Generate the IMG disk image using the dd command
        const createImgImage = (sourceDir: string, outputPath: string) => {
          return new Promise<void>((resolve, reject) => {
            exec(`dd if="${sourceDir}" of="${outputPath}"`, (error, stdout, stderr) => {
              if (error) {
                reject(error);
              } else {
                resolve();
              }
            });
          });
        };

        try {
          await createImgImage(tempDir, outputImgPath);
          returnData.push({
            json: {
              tarArchivePath,
              outputImgPath,
              status: 'success',
            },
          });
        } catch (error) {
          returnData.push({
            json: {
              tarArchivePath,
              outputImgPath,
              status: 'error',
              error: error.message,
            },
          });
        } finally {
          // Clean up the temporary directory
          fs.rmSync(tempDir, { recursive: true, force: true });
        }
      }

      return this.prepareOutputData(returnData);
    }
}
```

Register the new node in n8n.nodes.ts:

```
import { ImgImageFromTar } from './ImgImageFromTar.node';

export const NodeTypes = [
    // ...
    new ImgImageFromTar(),
];
```

Please note that the code provided here serves as a starting point for creating a custom n8n.io node. The actual logic to convert the extracted tar archive to an IMG disk image might need to be adjusted depending on the specific requirements and environment. The current implementation uses the dd command, which might not be suitable for all use cases. You may need to use external libraries or tools to achieve the desired result.