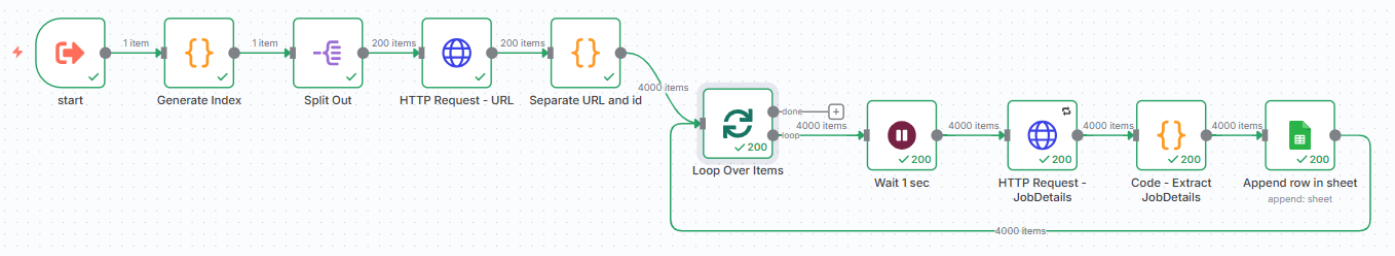
Workflow Documentation: MyCareersFuture Job Scraper

Objective: To automatically retrieve a complete list of all available jobs from the MyCareersFuture API (https://api.mycareersfuture.gov.sg/v2/jobs), extract relevant details from each job listing, and save the compiled data into a Google Sheet or a similar spreadsheet.

API Overview: The MyCareersFuture API is paginated. This means a single request does not return all jobs at once. Instead, it returns a limited number of jobs (e.g., 20-50 per page) along with a `total` count and the number of `pages`. The workflow below is designed to handle this pagination efficiently.



Workflow Step-by-Step Explanation

Here is a breakdown of each node in your workflow, explaining its purpose and configuration.

1. Start Node

- **Purpose:** This is the trigger that initiates the workflow. In n8n, this is typically a manual "Execute Workflow" button or could be scheduled (e.g., run daily at 9 AM). It has no configuration and simply passes an empty execution payload to the next node.

2. Generate Index Code Node

- **Purpose:** To create a static list of page numbers for the workflow to process, rather than dynamically calculating the number of pages from the API's response.
- **How it works:** This is a **Code** node that uses a simple JavaScript loop.
- **Code Analysis:**

```
// This loop runs 50 * 4 = 200 times
var noOfLoop = 50 * 4; // 200
for (let index = 1; index <= noOfLoop; index++) {
  list.push(index); // Adds numbers 1 through 200 to the list
}
return {list};
```

- **What it does:** It generates an array `[1, 2, 3, ..., 200]` and returns it. This means the workflow is hardcoded to process the first **200 pages** of the API results

3. Split Out Node

- **Purpose:** To take the single array output from the "Generate Index" node (which contains all page numbers) and split it into individual, separate items. This allows n8n to process each page number in parallel or sequence.
- **How it works:** This is an **Item Lists** node configured in "Split Out Items" mode.
  - It receives input from the "Generate Index" node as a single object: `{ "list": [1, 2, 3, ..., 200] }`.
  - Its job is to "unpack" this array. It looks at the `list` field and creates a new, separate output item for each value inside that array.

Configuration:

1. **Operation:** `Split Out Items`
2. **Field to Split:** `list` (This tells the node which field contains the array you want to split apart).
3. **Options:** `Include No Other Fields` (This is a crucial setting. It means the node will output *only* the individual page numbers. It will *not* include the original `list` array or any other data in the output, resulting in a clean data stream).

Transformation Example:

Input: `{ "list": [1, 2, 3, 4] }` (One item with a large array)
Output:

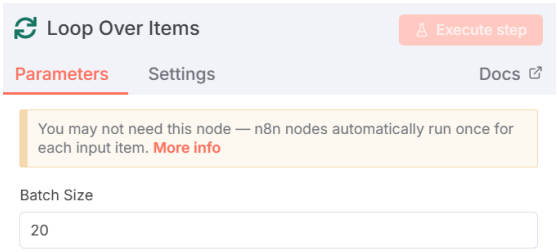
Item 1: `{ "list": 1 }`
Item 2: `{ "list": 2 }`
Item 3: `{ "list": 3 }`
Item 4: `{ "list": 4 }`
This creates 200 separate execution paths from the initial single one.
In summary: The "Split Out" node is the crucial link that takes the generated list of 200 page numbers and prepares them for the **HTTP Request** node. It converts one batch of data into many individual units of work, enabling the workflow to loop through each API page request efficiently. The `Include No Other Fields` option ensures the HTTP Request node receives a clean `{{$json.list}}` value (which is just a single number) to use in its URL.

4. HTTP Request Node (URL)

- **Purpose:** To perform the actual API call to MyCareersFuture for a specific page number.
- **How it works:** This is an **HTTP Request** node. Its key configuration is its URL. It uses the `page` value from the previous node's output to build a dynamic URL.
- **Configuration Example:**
  - **Method:** `GET`
  - **URL:** `https://api.mycareersfuture.gov.sg/v2/jobs?page={{$json.page}}&limit=50`
  - The `{{$json.page}}` is an n8n expression that gets replaced by the current page number (e.g., 1, 2, 3...) from the incoming data stream.
- **Output:** The full JSON response from the API for that page, which includes an array of job summary objects under a key like `results`.

5. Loop Over Items Node

- **Purpose:** To take the array of job summaries from the first HTTP Request node and split them into individual job items. This is necessary because the next step (the second HTTP Request) needs to request details for each job individually using its unique ID.
- **Batch Size:** 20



- Setting a batch size tells n8n to process items in parallel, significantly speeding up the workflow.
- **HTTP Request - URL:** Makes 200 separate API calls (one for each page). Assume each page returns 20 job summaries.
  - **Total Job Summaries:** 200 pages * 20 jobs/page = 4000 jobs
- **Separate URL and Id & Loop Over Items:** This is where the batch size kicks in. The "Loop Over Items" node receives a stream of data containing **4000 individual job IDs**.
- The node does not process these 4000 items all at once. Instead, it groups them into "batches" based on the **Batch Size** setting.
  - **Batch Size = 20** means the node will take the first 20 job IDs and send them to the next node (**HTTP Request - JobDetails**) to be processed in parallel.
  - Once that batch of 20 is finished, it takes the next 20, and so on.

The workflow will process the 4000 items in a series of **200 batches**:

- **Batch 1:** Jobs 1 - 20
- **Batch 2:** Jobs 21 - 40
- **Batch 3:** Jobs 41 - 60
- ....
- **Batch 199:** Jobs 3961 - 3980
- **Batch 200:** Jobs 3981 - 4000

Why This is Necessary (The Real Purpose)

Processing 4000 API calls in parallel is neither possible nor desirable. The batch size acts as a crucial control mechanism:

1. **Prevents Overwhelming the API:** Making 4000 simultaneous requests to `mycareersfuture.gov.sg` would almost certainly be identified as a Denial-of-Service (DoS) attack and get the IP address blocked. The batch size throttles the requests to a manageable level.
2. **Avoids System Overload:** Processing a huge number of items in parallel consumes a lot of memory and CPU on the machine running n8n. Batching prevents the workflow from crashing the server or n8n instance.
3. **Manages Rate Limits:** Most APIs, including MyCareersFuture, have rate limits (e.g., "X requests per minute"). Next, pair with a **"Wait" or "Delay" node** between batches to stay within the allowed limits.

6. Wait Node

- Configure it to wait for a short period (e.g., 100-500 milliseconds) between initiating each request in the batch. This further "smoothes out" the burst of traffic.

7. HTTP Request Node (JobDetails)

- **Purpose:** To fetch the detailed information for a single job. The initial list endpoint often returns a summary of jobs, while a dedicated detail endpoint provides comprehensive data (e.g., full job description, specific skills, application process).
- **How it works:** This is a second **HTTP Request** node. It uses a unique job identifier (like a `UUID` or `id`) from the 'Separate URL and id' node to construct the URL for the detailed endpoint.
- **Configuration Example:**
  - **Method:** `GET`
  - **URL:** `https://api.mycareersfuture.gov.sg/v2/jobs/{{$json.uuid}}` (The `uuid` field comes from the individual job item passed into this node).

8. Extract JobDetails Code Node

- **Purpose:** To transform the data and extract only the necessary fields from the large JSON object returned by the previous HTTP Request Node. This step cleans and structures the data for easy insertion into a spreadsheet.

9. Append row in sheet Node (Google Sheets)

- **Purpose:** To save the extracted job data from the previous node into a row in a spreadsheet.
- **How it works:** This is a **Google Sheets** node (or similar, like Airtable) configured in **"Append"** mode. It takes the clean JSON object from the "Extract Job Details" step and maps each field to a column in the Google Sheet.
- **Configuration:**
  - **Operation:** `Append`
  - **Sheet ID:** Google Sheet ID.
  - **Columns:** You define the column names exactly as they are in the sheet (e.g., "Job Title", "Company", "Skills").
  - The node automatically maps the incoming JSON properties (e.g., `job_title`) to the columns with matching names.

Workflow Summary & Flowchart

This workflow efficiently handles a paginated API through a nested loop structure:

1. **Outer Loop (Pages):** Generates all page numbers, then loops through them to request each page of job summaries.
2. **Inner Loop (Jobs per Page):** For each page of summaries, it loops through each individual job to request its full details.
3. **Data Processing & Storage:** For each job's full details, it extracts specific fields and appends them as a new row in a spreadsheet.

Notes

The two HTTP Request nodes serve **two completely different purposes**.

They are needed because the MyCareersFuture API (and many other modern APIs) typically has a two-tier structure for delivering data:

1. A **List/Summary Endpoint** (the first HTTP Request)
2. A **Detail Endpoint** (the second HTTP Request)

Here is a detailed breakdown of why both are necessary.

1. First HTTP Request: The "URL" Node

- **Purpose:** To fetch a **paginated list of job summaries**.
- **API Endpoint:** `GET https://api.mycareersfuture.gov.sg/v2/jobs`
- **What it Returns:** When you call this endpoint, you do not get all the information about every job. Instead, you get a single **page** of results, where each job is represented by a summary object, as seen below:

- **Why it's needed:** This node is crucial for **discovery**. Its job is to quickly get the unique identifiers (like the **jobDetailsUrl** the example above) and basic info for all jobs on a given page.

2. Second HTTP Request: The "JobDetails" Node

- **Purpose:** To fetch the **complete, detailed information for one specific job**.
- **API Endpoint:** `GET https://api.mycareersfuture.gov.sg/v2/jobs/{uuid}` (where `{uuid}` is the unique ID from the first request)
- **What it Returns:** This call returns a very rich, detailed object for the single job specified by the UUID.
- The response would be much more comprehensive:


```
[
  {
    "uuid": "14f51bb7ebdb120cb9fa1b6de2275af2",
    "sourceCode": "Employer Portal",
    "title": "Administrator (Dental Laboratory)",
    "description": "<ul>\n  <li><strong>Well
                    Established Company</strong>
                    </li>\n  <li><strong>Up to $2800 +
                    AWS + Variable Bonus</strong>
                    </li>\n  <li><strong>Excellent
                    Welfare and Benefits + Career
                    progression</strong></li>\n  <li>
                    <strong>Working location:
```


- **Why it's needed:** This node is crucial for **data extraction**. The data from this request is what gets processed and sent to the spreadsheet.

Output