Automatiser sa réduction de données spectroscopiques sans coder avec ARP Lite (version macOs)

En astronomie, le traitement de données spectroscopique est une action relativement répétitive. Qui dit répétitif, dit donc théoriquement automatisable. Je vous propose au travers de cet article de tenter une approche rapide et sans code de cette automatisation.

Lorsque l'on est astronome amateur, que l'on pratique l'astrophotographie ou la spectroscopie, les données s'accumulent vite et en masse. Outre la nécessité de les organiser intelligemment pour s'y retrouver, il est souvent utile de les traiter peu de temps après l'acquisition pour se rappeler d'éventuelles particularités et garder la session en tête. Automatiser ces traitements permet de gagner du temps afin de se concentrer sur l'analyse des données.

Plutôt que de fournir un logiciel ou une solution clé en main, je vous propose de construire ensemble un processus qui peut s'adapter à vos contraintes et usages, du moins je l'espère. Cela sans taper une ligne de code, tout en permettant d'en ajouter à l'envie.

En effet, depuis plusieurs années maintenant, une tendance se développe et se fait une petite place dans les manières de mettre en œuvre des applications, la tendance No Code.

No Code

Cette pratique du No Code vise à assembler des briques logicielles préprogrammées et configurables en vue de concevoir des logiciels, comme des applications mobiles ou des processus, mais sans avoir à coder, vous l'aurez compris. Cette solution ne répond pas à tous les besoins, cependant il est parfois tout de même possible d'ajouter ou d'exécuter du code en complément. Je vous propose donc d'utiliser une de ces solutions pour construire notre processus, en « no code ».

Acquisition

Ici, nous partirons du principe que la solution d'acquisition est laissée aux choix de l'astronome. Outre l'avantage de permettre à chacun d'utiliser la solution d'acquisition qu'il ou elle préfère, cela permettra aussi demain de garder tel quel le processus de réduction automatique si nous venons à changer de logiciel d'acquisition.

Plusieurs logiciels permettent effectivement d'acquérir des spectres aisément tels que Prism, ou MaximDL, qui autorisent aussi d'automatiser les acquisitions par des scripts complémentaires. Ici, j'ai utilisé le protocole de communication distribué INDI associé au planétarium KStars, déjà présenté dans un autre article : https://stellartrip.net/spectre-de-phecda/

Pour illustrer notre processus, partons d'une observation effectuée avec le spectroscope Star'Ex en impression 3D, mis en œuvre par C. Buil (voir page du projet ici : https://astrosurf.com/solex/starex).

Le cœur du processus de réduction peut ainsi être réalisé avec le logiciel SpecINTI qui a besoin de deux fichiers :

- Le fichier décrivant l'observation et les fichiers résultants.

```yaml
Alkaid:
- Alkaid-        # Nom catalogue
- Alkaid-        # Noms fichiers objet
- 7              # Nombre de fichiers objet
- Alkaid_neon    # Nom fichier étalonnage
- 7              # Nombre
- Flat-          # ...
- 7
- Dark-
- 4
- o-
- 7
- None
- 0
```

- Le fichier décrivant la configuration matérielle et les options de traitement que nous souhaitons appliquer.

```yaml
# ******************************************************************************
# Configuration Star'Ex  2400 t/mm - 80x125 - fente 19 microns - Mak127
# Etalonnage en mode 3
# ******************************************************************************

# ---------------------------------------------------
# Répertoire de travail
# ---------------------------------------------------
working_path: /Users/matthieu/lab/arp_lite/last_session/  # ne pas oublier le / à la fin

# ---------------------------------------------------
# Fichier batch de traitement
# ---------------------------------------------------
batch_name: last_obs


# ---------------------------------------------------------------
# Mode d'étalonnage spectral
# ---------------------------------------------------------------

calib_mode: 3

# ---------------------------------------------------------------
# Recherche automatique des raies d'étalonnage
# ---------------------------------------------------------------
auto_calib: [6500, 6700]


# ---------------------------------------------------------------
# Largeur en pixels de la zone de recherche des raies
# d'étalonnage spectral
# ---------------------------------------------------------------
search_wide: 100
```

Nous reviendrons sur ces fichiers et leurs configurations un peu plus tard.

Cet exécutable s'accompagne d'un dossier nommé _configuration qui se place au même endroit dans lequel nous allons placer le fameux fichier décrivant le matériel et le traitement. Pour plus d'informations sur le logiciel et son fonctionnement, je ne saurai que vous conseiller la page du projet de C. Buil qui détaille très bien le tout, à cette adresse : [https://astrosurf.c….] http://www.astrosurf.com/solex/specinti.html .

Ce moteur de réduction SpecINTI existe en différentes versions exécutables sur les systèmes d'exploitation les plus courants. Cela permet ainsi d'utiliser cette solution d'automatisation sur Windows, MacOs, Linux, et également sur Raspberry Pi.

Quoi de mieux qu'une carte Raspberry Pi, compacte et peu consommatrice d'énergie pour cela ?

Automatisation

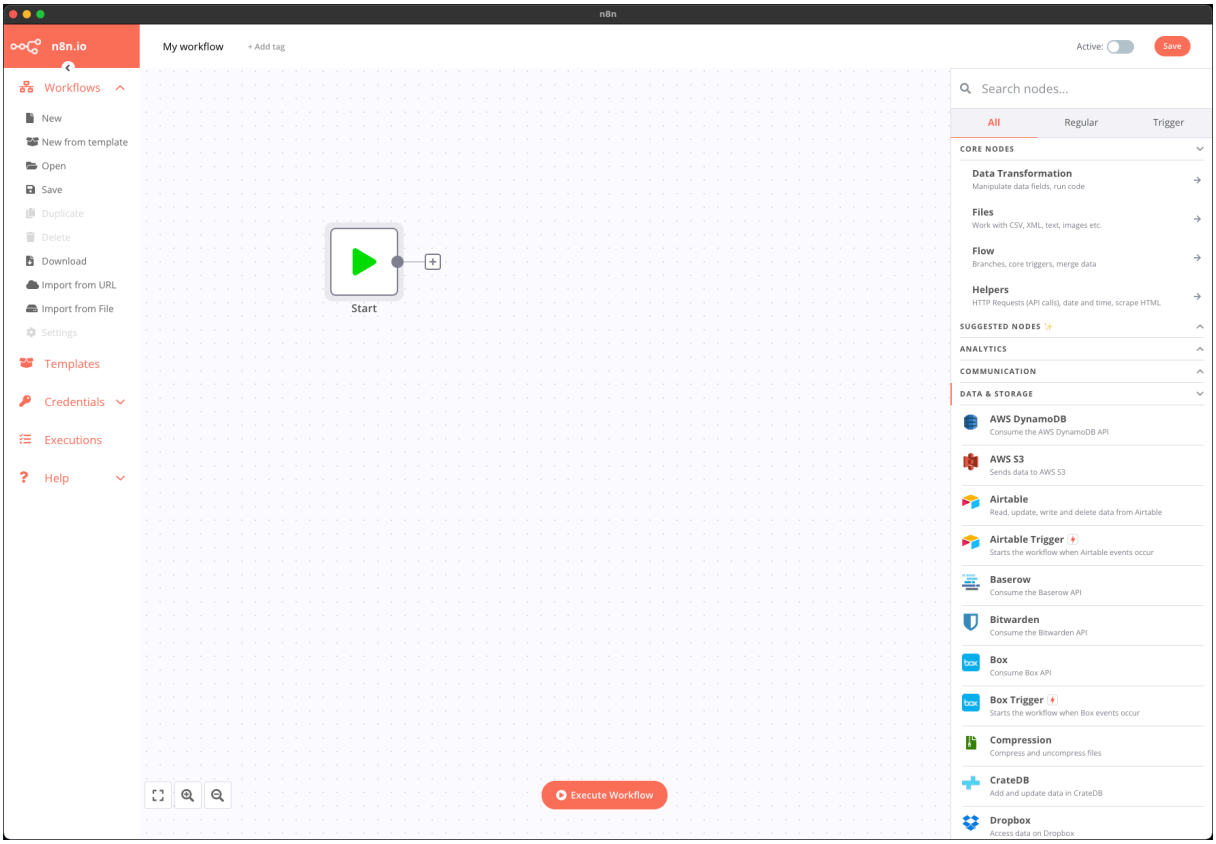
Maintenant que nous avons tout ce qu'il nous faut, nous pouvons dérouler le processus qui s'appliquera à chaque fois qu'une session sera à traiter.

n8n

Pour mettre en œuvre ce processus, nous utiliserons ainsi l'outil nommé n8n (que l'on prononce n - eight - n) qui est un outil d'automatisation de flux de travail basé sur des nœuds.

Site officiel : https://n8n.io/

Cet outil dispose d'une interface vraiment facile et agréable à utiliser grâce à laquelle nous pouvons, par simple glisser-déposer, construire nos différentes étapes.



Les nœuds correspondent à différentes étapes de calcul, de vérification ou connexion auprès de services que nous pouvons configurer par exemple. De nombreux types de nœuds existent déjà, que ce soit des d'opération simple comme la lecture de fichiers CSV, la récupération de données auprès d'API comme celle de la NASA, mais aussi une foule de services en ligne comme Dropbox, Google Drive et j'en passe.

Search nodes...

All | Regular | Trigger

CORE NODES

Data Transformation
Manipulate data fields, run code

Files
Work with CSV, XML, text, images etc.

Flow
Branches, core triggers, merge data

Helpers
HTTP Requests (API calls), date and time, scrape HTML

SUGGESTED NODES

Execute Command
Executes a command on the host

Function
Run custom function code which gets executed once and allows you to add, remove, change and replace items

HTTP Request
Makes an HTTP request and returns the response data

Set
Sets values on items and optionally remove other values

Switch
Route items depending on defined expression or rules

Webhook
Starts the workflow when a webhook is called

ANALYTICS

Cortex
Apply the Cortex analyzer/responder on the given entity

Google Analytics
Use the Google Analytics API

Google Cloud Natural Language
Consume Google Cloud Natural Language API

Google Perspective
Consume Google Perspective API

Grafana
Consume the Grafana API

Search nodes...

All | Regular | Trigger

BambooHR
Consume BambooHR API

Customer Datastore (n8n training)
Dummy node used for n8n training

Customer Messenger (n8n training)
Dummy node used for n8n training

DHL
Consume DHL API

Figma Trigger (Beta)
Starts the workflow when Figma events occur

Google Books
Read data from Google Books

Google Contacts
Consume Google Contacts API

Google Docs
Consume Google Docs API.

Home Assistant
Consume Home Assistant API

LingvaNex
Consume LingvaNex API

NASA
Retrieve data from the NASA API

Onfleet
Consume Onfleet API

Onfleet Trigger
Starts the workflow when Onfleet events occur

OpenWeatherMap
Gets current and future weather information

Philips Hue
Consume Philips Hue API

Spotify
Access public song data via the Spotify API

Workable Trigger
Starts the workflow when Workable events occur

Trois modes d'installation sont possibles :

1. Installation de l'application sur sa machine personnelle (Windows, MacOS, et bientôt Linux). Ce mode est gratuit.
2. Installation de la solution manuellement autrement dit autohébergée, par exemple sur Raspberry. Pi ! Ce mode est gratuit.
3. Utilisation de la solution en mode cloud, en ligne. Ce mode est payant.

Pour que les étapes décrites ici puissent être reproduitent facilement et par le plus grand nombre, **nous utiliserons la première solution, à savoir l'application bureautique.**

Peu de choses diffèrent entre les systèmes d'exploitation concernant l'usage de l'application, il s'agit surtout de chemins différents ou de commandes propres au système. Cependant, pour répondre aux usages les plus larges, j'ai écrit ce même article en fonction du système d'exploitation sur lequel vous utiliserez l'application.

La version en ligne, que vous lisez actuellement, se base sur un système MacOS qui s'appliquera aussi sur Linux, mais une version PDF pour Windows est disponible ci-après. Cela vous permettra de suivre cet article pour découvrir la solution ensemble, mais également de tester sur votre machine ensuite.

Par ailleurs, la totalité des fichiers d'observation utilisés ainsi que le processus que nous allons construire sont téléchargeable à la fin de cette article.

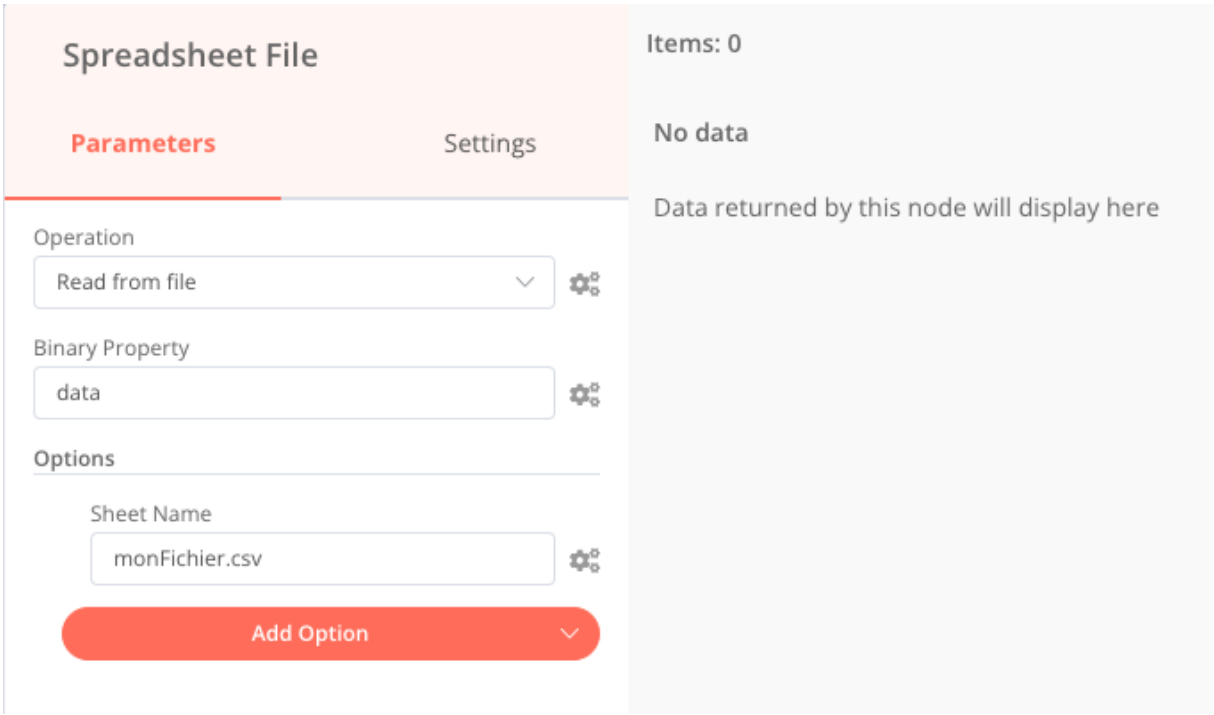
Allons-y. Rendez-vous sur le site officiel de n8n, par ici : https://n8n.io/, sélectionnez *Desktop app* en bas de page et suivez les instructions d'installation classiques.

Premiers pas sur n8n

Nous avons ainsi maintenant notre application ouverte, sur le bureau. À gauche se trouve le menu de gestion des *Workflows* avec les opérations classiques, nouveau, ouvrir, sauvegarder, …

Le centre de la fenêtre va contenir toutes les étapes graphiques de notre processus. Lors de la création d'un nouveau processus, un item *Start* est affiché. À droite se trouvent les fameux nœuds que nous allons utiliser, il nous suffit de cliquer sur un des noeuds pour l'ajouter à l'espace central et les raccorder ensemble pour les utiliser.

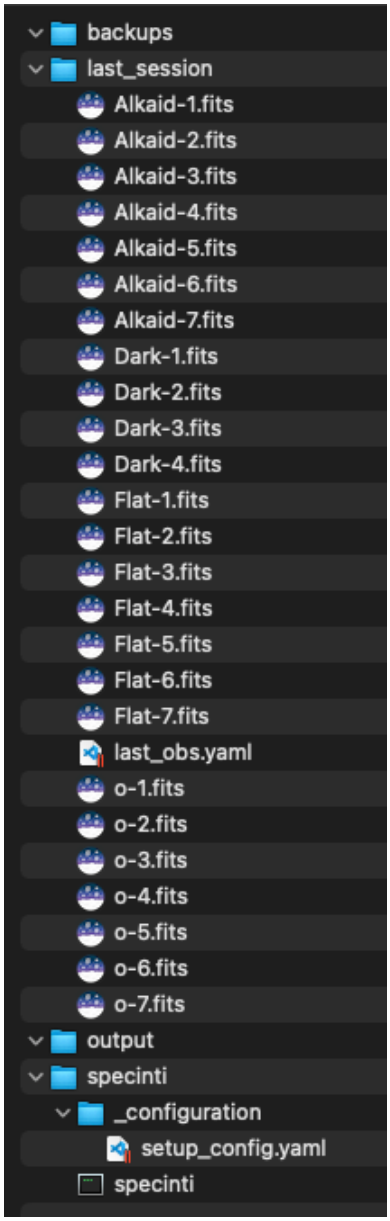
Une fois ajouté, un double-clic sur le noeud permet d'accéder à sa configuration. Par exemple, le nœud Speadsheet File permet de lire ou écrire un fichier CSV ou XLS.



Je vous propose de poser maintenant les grandes lignes de notre flux de réduction automatique. Pour rester sur un usage simple, nous prendrons l'exemple d'un ob-

servatoire dans lequel se trouve un ordinateur piloté à distance qui contrôle le matériel et qui contient les fichiers de la dernière session enregistrée la veille. On part donc du principe que l'application n8n est donc installé sur cet ordinateur.

Sur cette ordinateur, nous avons également un dossier de travail dans lequel se trouvent les fichiers d'observations et tout ce dont nous aurons besoin pour cette automatisation. Voici l'arborescence globale de ce dossier de travail :



- Un dossier **backups** dans lequel nous sauvegarderons nos fichiers de sessions avant chaque traitement pour éviter de perdre notre session si notre chaîne de traitement déraille…
- Un dossier **last_session** qui contient tous nos fichiers d'observations à traiter
- Un dossier **specinti** qui contient l'exécutable SpecINTI et le fichier de configuration matériel.
- Un dossier **output**

Les étapes, lors du déclenchement de notre *Workflow, seront :*

- Vérification de la présence de fichiers à traiter
- Sauvegarde des fichiers (une copie dans un autre dossier local)
- Lancement de la réduction avec SpecINTI
- Envoi d'une notification à l'astronome pour lui préciser la fin de traitement
- Sauvegarde de tous les résultats (une copie dans un autre dossier local)

Nous prendrons le temps de démarrer en détaillant les premières étapes, puis nous ajouterons les étapes plus rapidement ensuite.

Automatic Reduction Pipeline (Lite)

Vérification de la présence des fichiers à traiter

À chaque déclenchement du processus, nous allons vérifier si des fichiers sont présents, le cas échéant, le processus s'arrête.

Pour faire cela, nous pouvons utiliser une commande pour lister les fichiers présents dans un dossier et les compter. Sur MacOS, cette commande terminal (bash) est la suivante :

```
ls nom_du_chemin | wc -l
```

Voici ce qu'elle fait cette commande :

- Liste tous les fichiers du dossier

  ```
  ls nom_du_chemin
  ```

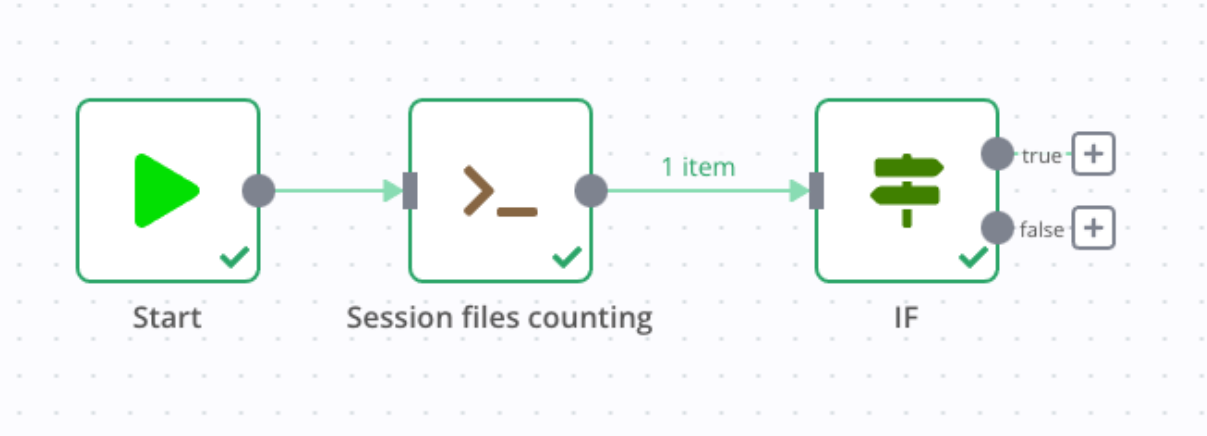
- Redirection de la liste des fichiers vers une autre commande grâce au *pipe (|)*
- Compte le nombre de mots/lignes/caractères. L'option -l permet ainsi de préciser que nous souhaitons compter les lignes, à savoir tous les fichiers présents.

  ```
  wc -l
  ```

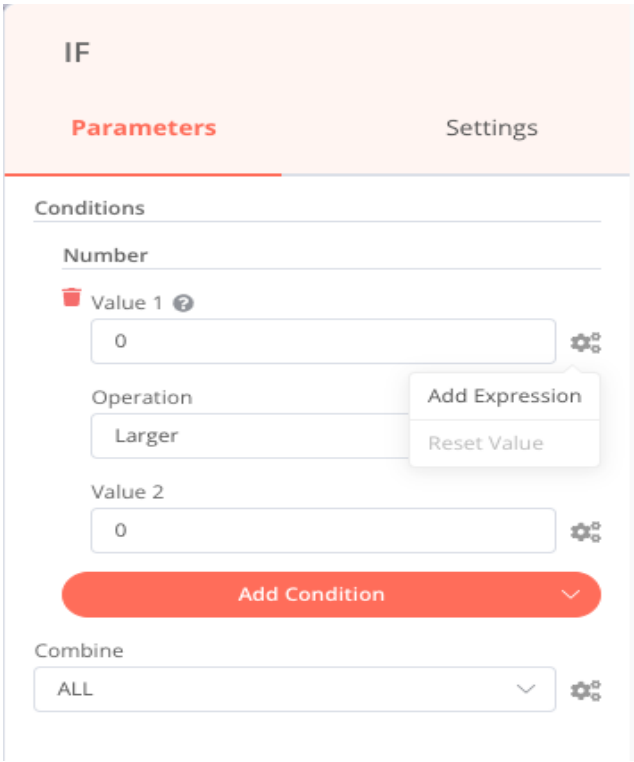
Pour faire cela, j'ajoute le nœud qui s'appelle *Execute Command*, dans lequel je vais intégrer la commande précédente. Lors de cette configuration, nous pouvons déjà voir la sortie sur le panneau de droite lors d'une exécution de test. Ici le package contient 26 fichiers que l'on retrouve dans la colonne *stdout*. Tout est bon, je peux le raccorder au premier nœud Start.

Maintenant, nous pouvons récupérer cette sortie au sein d'un nœud suivant et nous en servir comme condition, avec le bloc *if.* Si des fichiers sont présents, on continue, sinon on arrête.

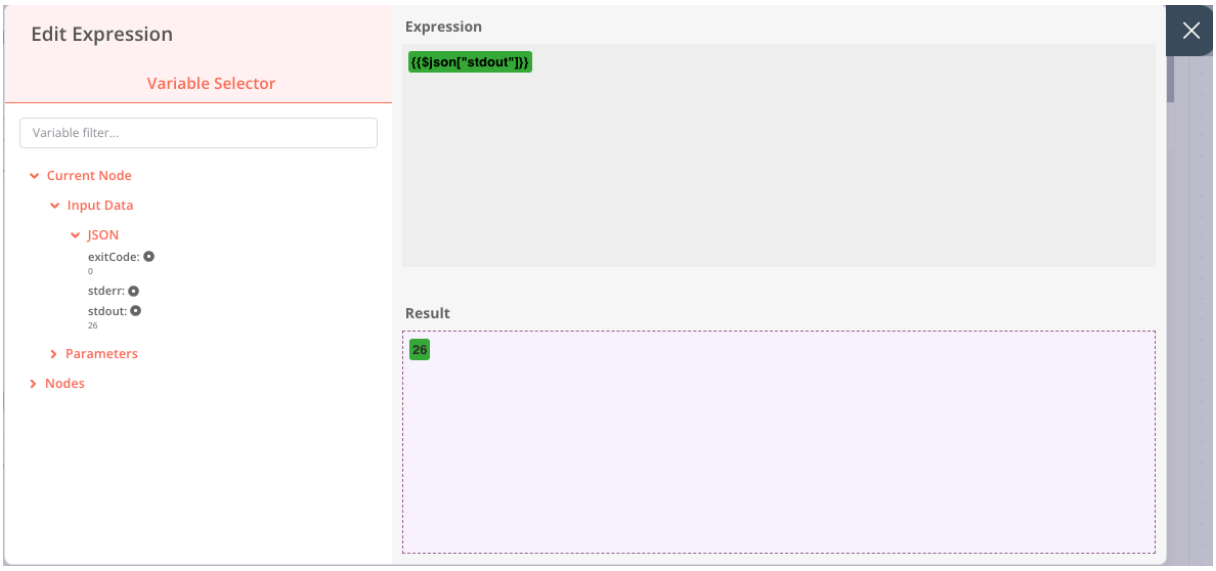
Dans un premier temps, raccorder le bloc *if* au noeud précédent sans le paramétrer et exécuter entièrement la chaîne. Cela va nous permettre de récupérer graphiquement le nombre de fichiers comptés précédemment.



Une fois cela fait, allons le paramétrer en double-cliquant sur ce noeud *IF*, puis en ajoutant une condition *Number*. Dans le premier champ de valeur, cliquez sur la petite roue dentée pour ajouter une expression, comme ci-dessous :

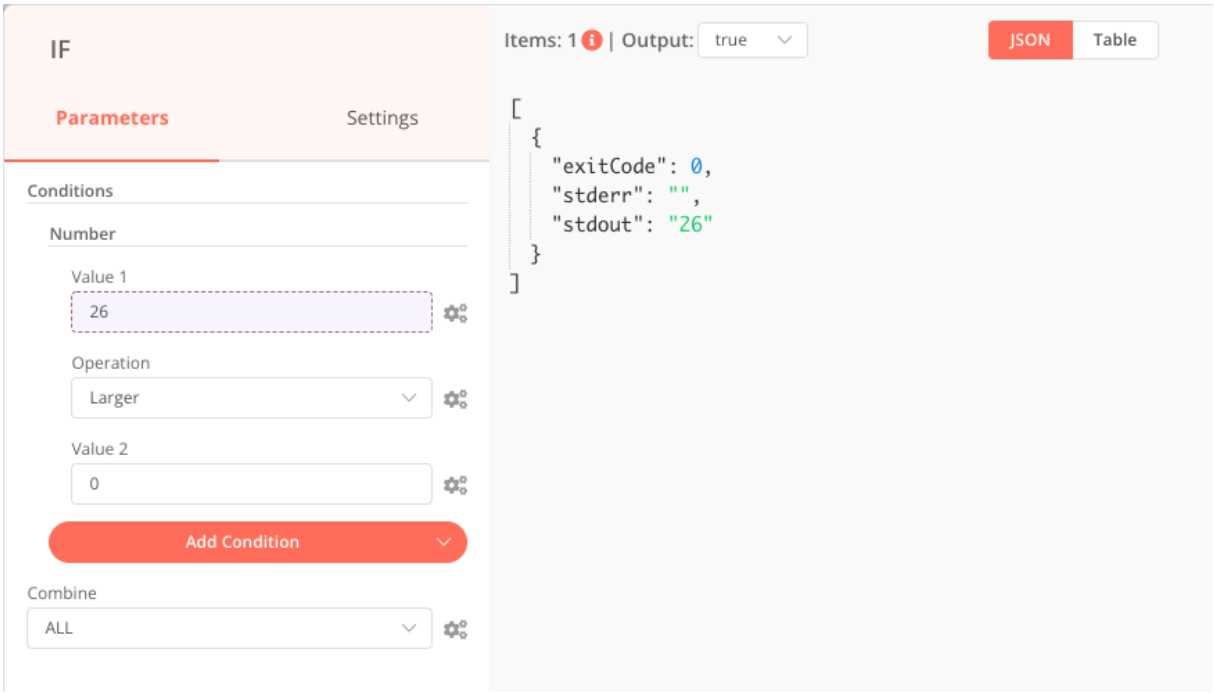


Une nouvelle fenêtre s'ouvre, et nous pouvons aller récupérer la valeur de ***stdout*** dans les inputs. En cliquant une seule fois, l'élément est ajouté. Si nous n'avions pas effectué l'exécution précédente, la valeur en entrée n'aurait pas été présente dans la liste car pas encore connue. Mais il est tout de même possible de l'ajouter manuellement dans le champs de droite.



Les autres champs de conditions sont configurés de sorte que si *stdout*, que nous venons de récupérer, est supérieur à 0, alors des fichiers sont présents et la sortie sera à True.

À noter ici qu'il s'agit d'une vérification assez légère, nous pourrions très bien avoir simplement plusieurs fichiers de texte et la condition serait validée. Libre à vous bien entendu de renforcer cette vérification ensuite.



Notre étape de vérification de présence de fichiers est complète.

Sauvegarde des fichiers

Pour sauvegarder ce dossier de fichier, là encore nous pouvons utiliser la facilité d'usage des commandes de notre système, en effectuant une simple copie dans un autre dossier. Mais on peut également très bien utiliser un nœud *Google Drive* , *Dropbox*, ou le nœud *FTP* pour sauvegarder nos fichiers ailleurs.

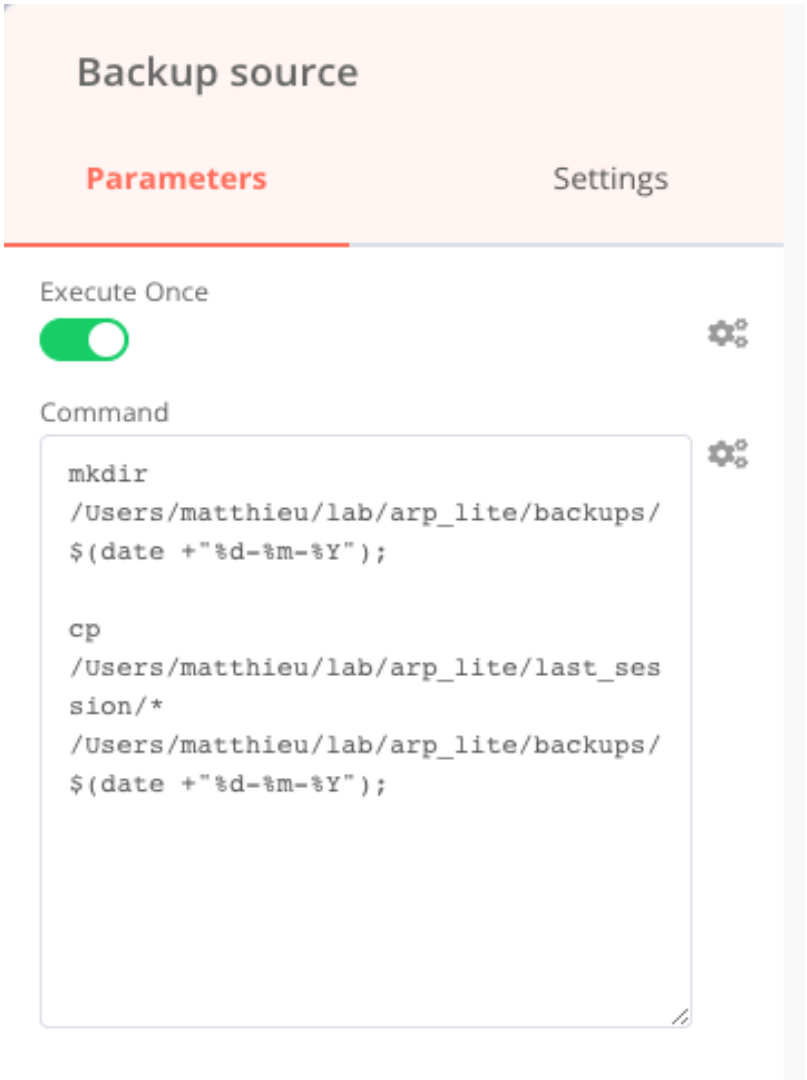
Pour faire cette copie, je vais créer un dossier en lui donnant le nom de la date de traitement à l'intérieur du dossier Backups. Pour faire cela, j'ajoute les deux commandes suivantes à la suite dans le même bloc *Execute commands*. Bien sûr, il faut remplacer le chemin du dossier last_session et backup par les vôtres.

Création d'un dossier ayant comme nom la date du jour dans backups

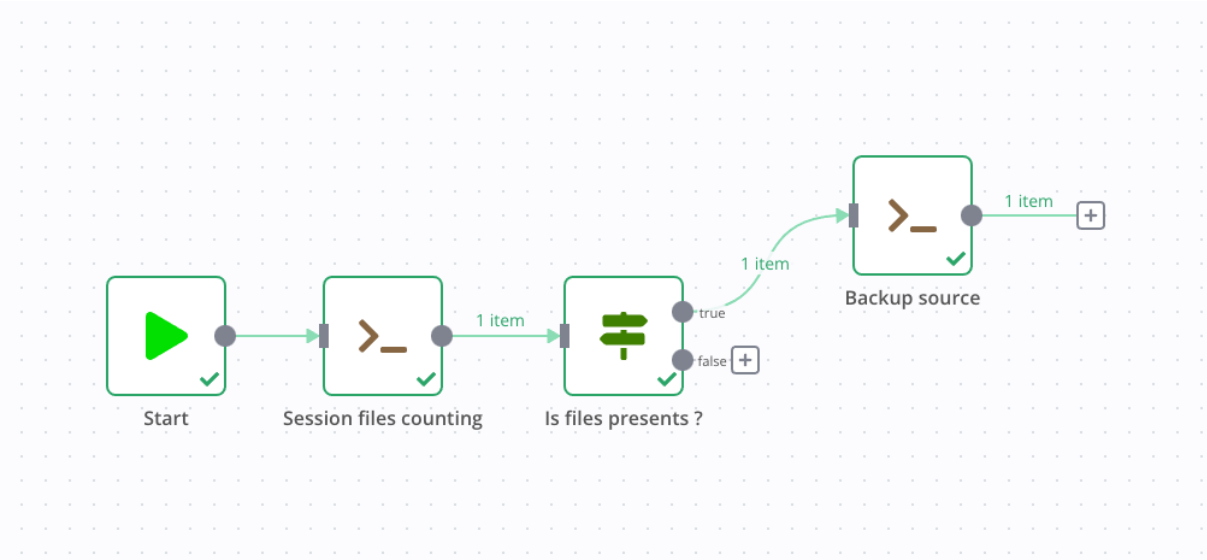
```
mkdir /Users/matthieu/lab/arp_lite/backups/$(date +"%d-%m-%Y");
```

Copie de tous les fichiers du dossier last_session dans le dossier backups/date-du-jour

```
cp /Users/matthieu/lab/arp_lite/last_session/*
/Users/matthieu/lab/arp_lite/backups/$(date +"%d-%m-%Y");
```



Nous pouvons raccorder ensuite ce nœud à la sortie True de notre nœud *If* précédent, cette étape est finie.



Réduction avec SpecInti

Nous sommes prêts pour lancer la réduction. SpecINTI est disponible sous forme d'exécutables, qu'il est possible de lancer par une simple commande système. Ça tombe bien, on commence à bien connaître le nœud *Execute command* . Rassurez-vous, nous verrons d'autres nœuds juste après !

J'ai donc téléchargé l'exécutable SpecINTI pour MacOS via cette page : https://stellartrip.net/specinti-mac-linux/, puis je l'ai placé dans le dossier arp_lite/specinti/ , ainsi que le fichier de configuration matérielle et de traitement dans le dossier _configuration

Points importants, notre fichier de configuration devra avoir toujours le même nom car il sera appelé à chaque traitement. Ici il est appelé setup*config.yaml*. Par ailleurs, les deux premiers paramètres à spécifier dans ce fichier de configuration sont :

- Le chemin des fichiers d'observations
- Le nom du fichier d'observation qui précisent les noms des fichiers fits. Comme nous allons automatiser le traitement, il s'avère important que le fichier d'observation ait toujours le même nom. Ici j'ai choisi de l'appeler **last_observation.yaml**

Nous avons déjà rencontré une vignette ce fichier au début de cet article, je le remet ici pour illustrer mon propos.

```yaml
# ******************************************************************************
# Configuration Star'Ex  2400 t/mm - 80x125 - fente 19 microns - Mak127
# Etalonnage en mode 3
# ******************************************************************************

# ---------------------------------------------------
# Répertoire de travail
# ---------------------------------------------------
working_path: /Users/matthieu/lab/arp_lite/last_session/  # ne pas oublier le / à la fin

# ---------------------------------------------------
# Fichier batch de traitement
# ---------------------------------------------------
batch_name: last_obs


# ---------------------------------------------------------------
# Mode d'étalonnage spectral
# ---------------------------------------------------------------

calib_mode: 3

# ---------------------------------------------------------------
# Recherche automatique des raies d'étalonnage
# ---------------------------------------------------------------
auto_calib: [6500, 6700]


# ---------------------------------------------------------------
# Largeur en pixels de la zone de recherche des raies
# d'étalonnage spectral
# ---------------------------------------------------------------
search_wide: 100
```

Rappel des règles d'automatisation :

- Si notre matériel ne change pas, le fichier de configuration ne change théoriquement pas. Le fichier de configuration sera toujours nommé **setup*con-fig.yaml***
- Les fichiers à traiter seront toujours dans le dossier **last_session**
- Le fichier d'observation sera toujours nommé **last_obs.yaml,**

Ajoutons donc un nœud *Execute Command* qui appelle SpecINTI et effectue cette réduction.

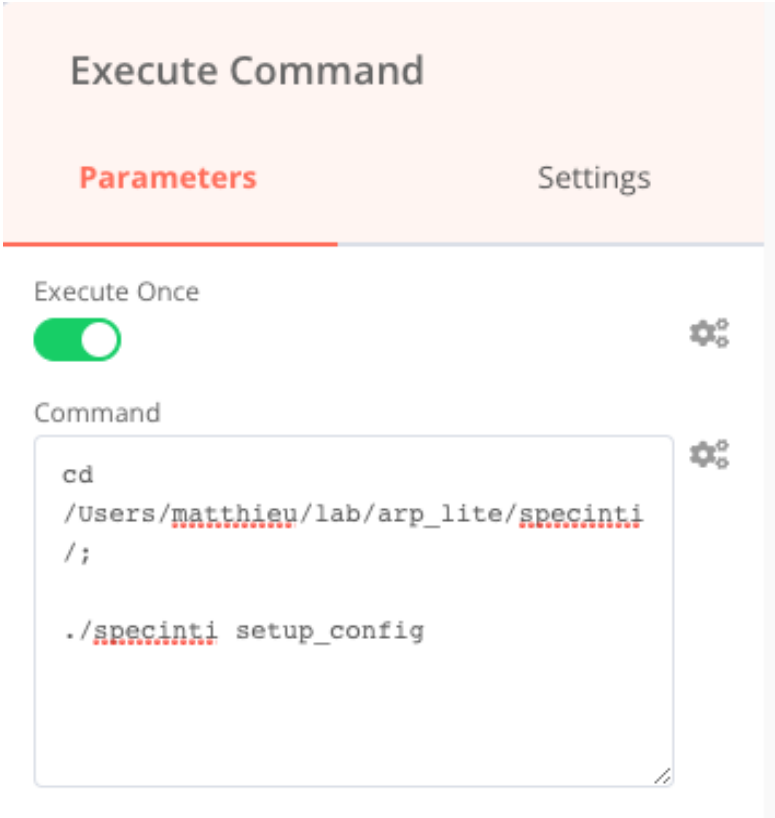
Déplacement dans le dossier specinti

```
cd /Users/matthieu/lab/arp_lite/specinti/;
```

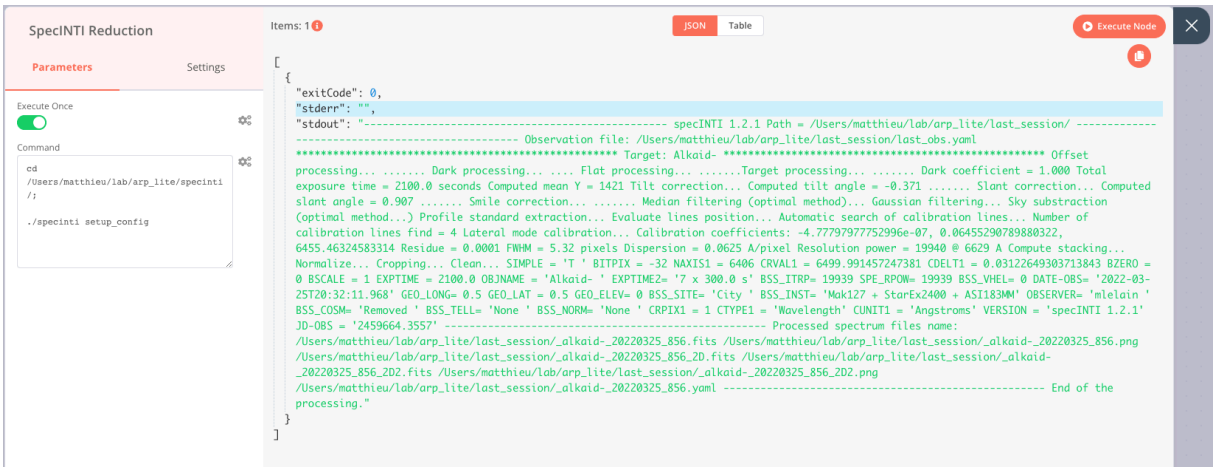
Lancement de SpecINTI avec le fichier de configuration en paramètre

```
./specinti setup_config
```

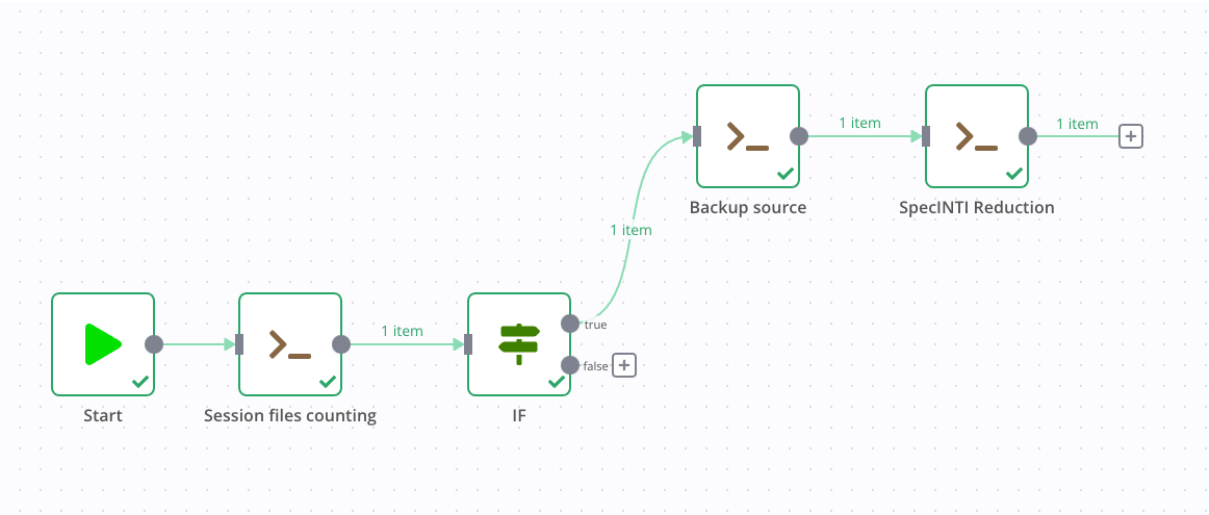
Et voilà !



À la suite de son exécution, ou lors d'un essai, on retrouve ainsi la sortie classique de SpecINTI sur stdout, avec les diverses informations du traitement qui s'affichent dans le terminal en temps normal.



Voici à quoi ressemble notre pipeline à ce stade.

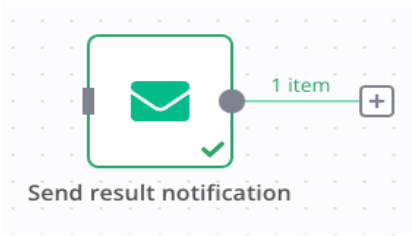


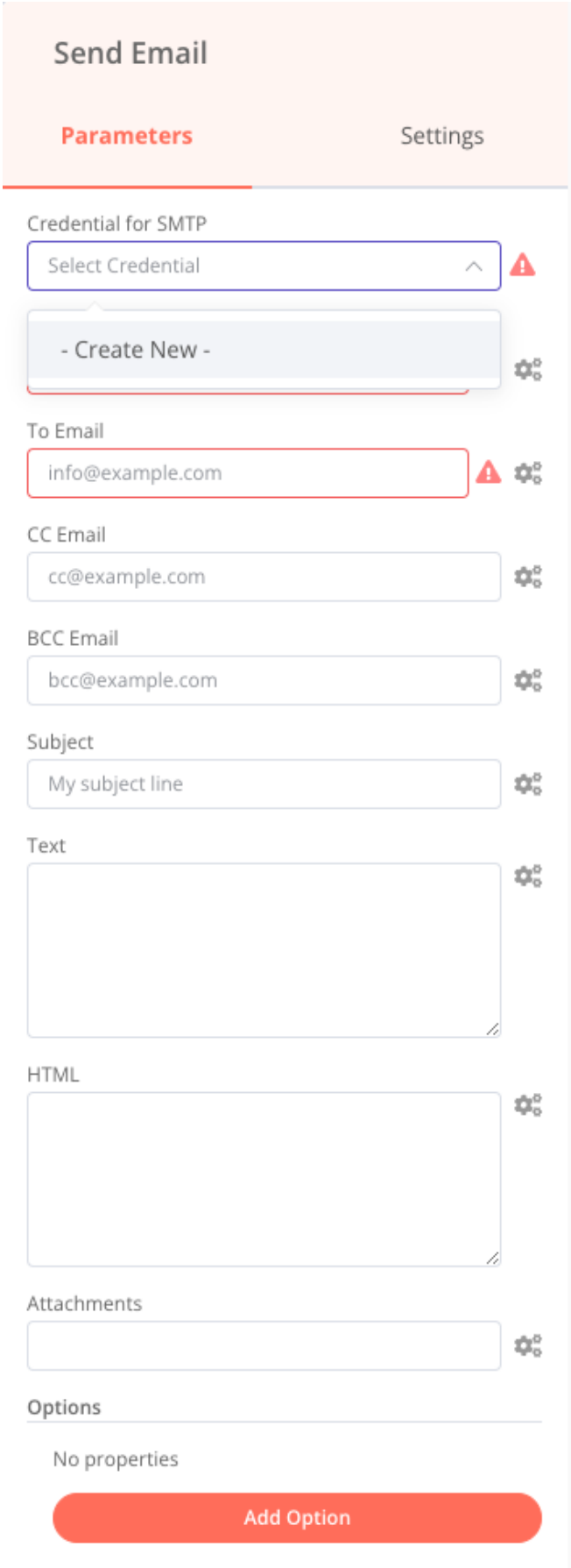
Notification de fin de traitement

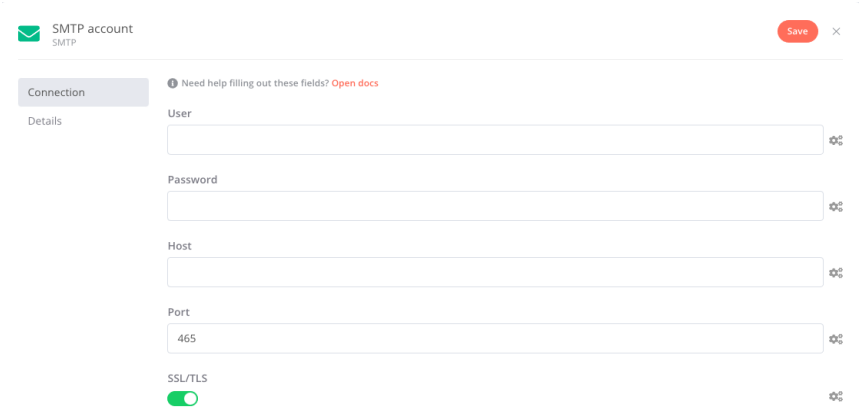
Maintenant que la réduction est terminée, nous pouvons ajouter un nœud pour prévenir l'astronome que ses données sont prêtes, par exemple avec le nœud *Send email*.

Mail

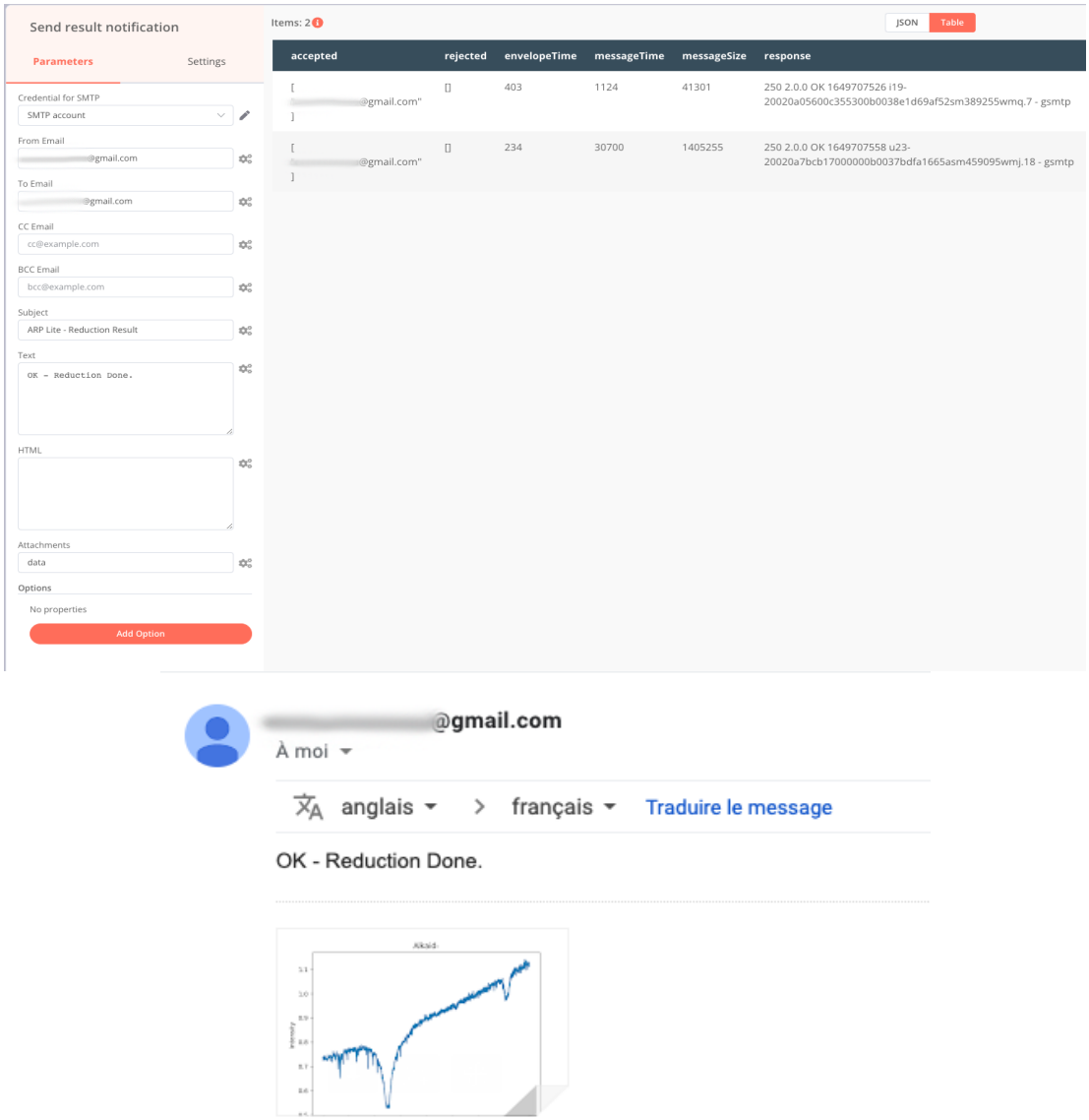
La première étape consiste a ajouter un compte e-mail pour pouvoir envoyer des mails, avec les informations classiques de connexion, adresse email, mot de passe, serveur hôte. Toutes les informations de connexions ne sont enregistrées que sur votre machine et sont ensuite accessibles dans le menu Credentials de l'application. Ajoutons ce noeud *Send Email*.



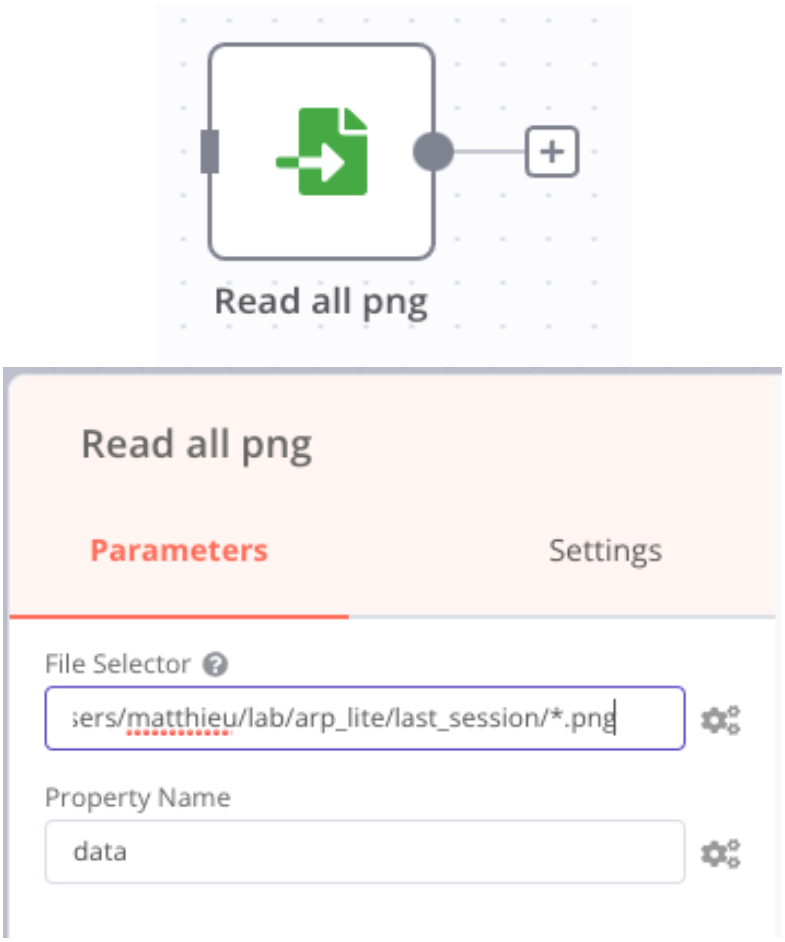

Send Email
Parameters
Settings
Credential for SMTP
Select Credential
- Create New -
To Email
info@example.com
CC Email
cc@example.com
BCC Email
bcc@example.com
Subject
My subject line
Text
HTML
Attachments
Options
No properties
Add Option



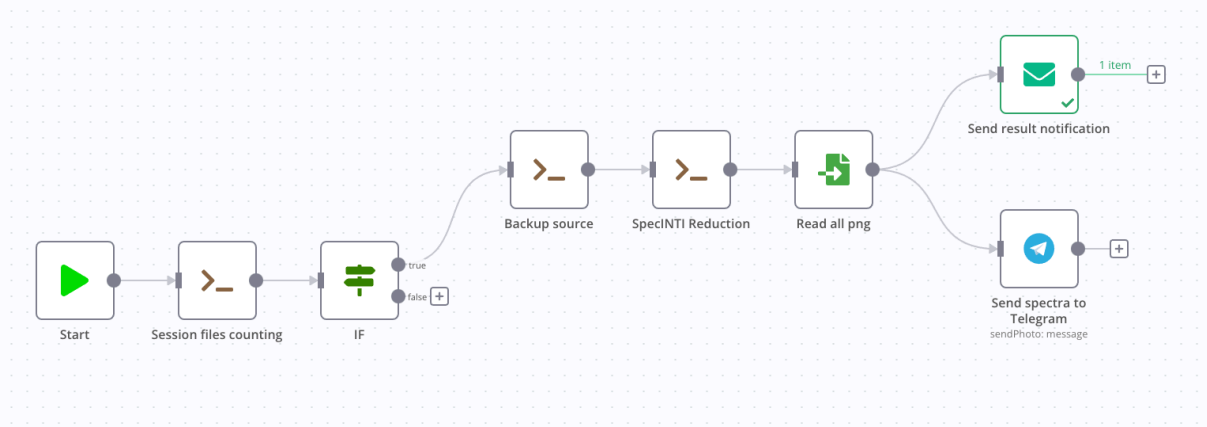
Dès lors, il reste ensuite à configurer le message que nous souhaitons envoyer vers une adresse de destination.



Nous pouvons également faire précéder le nœud *Send email* par un nœud *Read Binary Files* qui va lire tous les fichiers PNG du dossier de résultat et les envoyer vers la sortie nommée *data*, puis nous pouvons intégrer ces images (nommée data) dans le mail de notification comme sur la capture d'écran précédente (champs attachements).



La même logique peut être appliquée avec le nœud Telegram (application de messagerie instantanée comparable à WhatsApp) qui prend en entrée la liste des fichiers PNG reçue, et nous recevons ainsi un message directement dans une conversation. Voici ci-dessous le résultat de la notification mobile et l'état de notre processus.



Sauvegarde des résultats

Nous pouvons maintenant finaliser notre chaîne de traitement avec une sauvegarde des résultats. Pour faire simple, nous allons simplement déplacer les fichiers dans le dossier ***output***, au sein d'un nouveau dossier nommé avec la date du jour comme pour la première sauvegarde. Cette fois nous utiliserons la commande **move** dans un nœud *Execute Command*.

Note : Il est important de déplacer tous les fichiers plutôt que de faire une copie, car demain une nouvelle session sera peut-être à traiter. Nous devons donc vider le dossier last_session.

Création d'un nouveau dossier dans output avec la date du jour comme nom

```
mkdir /Users/matthieu/lab/arp_lite/output/$(date +"%d-%m-%Y");
```

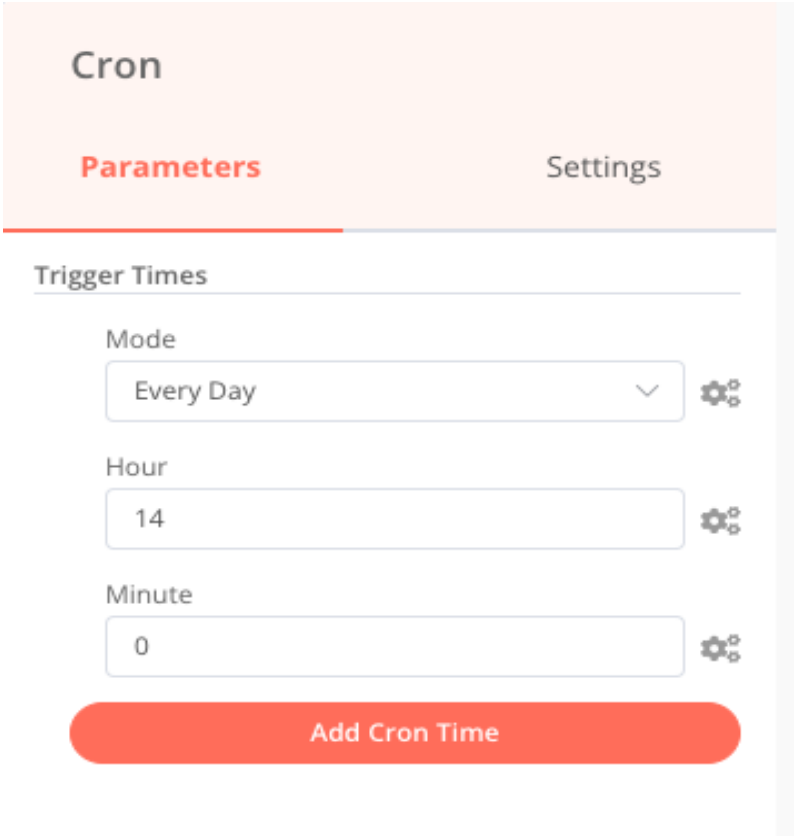
Déplacement de tous les fichiers du dossier last_session dans le dossier créé juste avant

```
 mv /Users/matthieu/lab/arp_lite/last_session/*
/Users/matthieu/lab/arp_lite/output/$(date +"%d-%m-%Y")/;
```

Pour faire les choses proprement, nous pouvons ajouter un nœud *Merge* précédemment qui va attendre la fin de l'envoi des deux notifications avant de lancer la sauvegarde. Nous pouvons également dupliquer l'envoi de mail pour prévenir l'astronome si le processus s'arrête au début lorsqu'il n'y a pas de fichiers à traiter.

Déclenchement

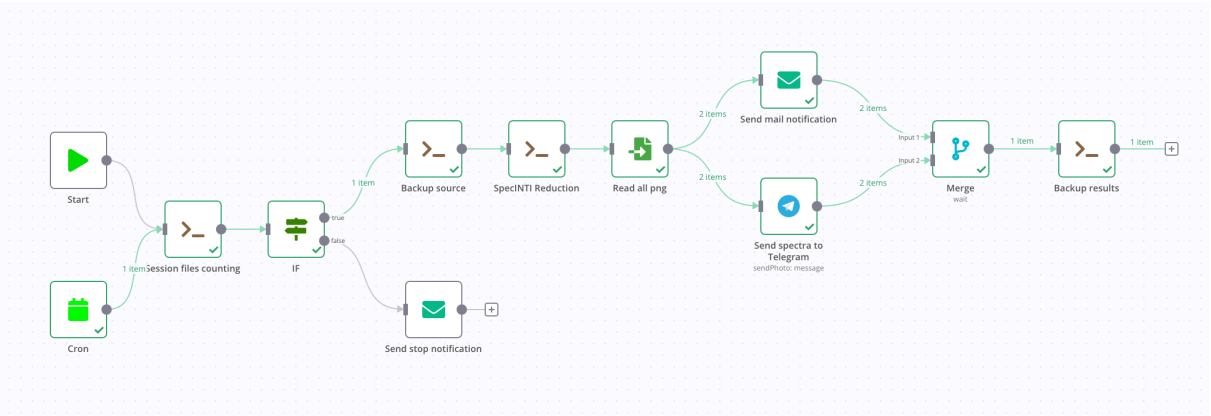
Notre chaîne de traitement est maintenant finalisée. Toutefois, elle ne se déclenche que manuellement, pas très automatique… Ajoutons un cron pour qu'elle se déclenche tous les jours vers 14H00.



Chaque jour, le traitement sera effectué si des fichiers sont présents, sinon il s'arrêtera. Nous avons aussi toujours la possibilité de le lancer manuellement grâce au nœud *Start* si l'attente de 14H est intenable !

Le nœud *Local files Trigger* est aussi une possibilité. Il peut être configurer pour vérifier les ajouts ou modifications de fichiers dans un dossier. Ainsi, nous pourrions très bien déclencher notre chaîne de traitement une fois toutes les acquisitions terminées en fin de nuit en ajoutant simplement une dernière image fits d'une seconde de pose avec un nom particulier qui servira de déclencheur, session_end.fits par exemple.

Cette fois c'est bon, notre pipeline de traitement automatique est terminé ! Voici le résultat graphique.



Vous trouverez également ci-dessous le code entier de ce processus au format json. Pour l'utiliser rien de plus simple, dans le menu de l'application, sélectionnez ***Import from file***, c'est fini. Il vous faudra tout de même modifier les nœuds avec vos propres chemins machines et informations de connexion mail si vous voulez le garder tel quel.

Je mets également le dossier entier avec les fichiers fits et les fichiers observation et configuration qui vous permettront d'avoir tout ce qu'il faut.

Liens dans l'article en ligne : https://stellartrip.net/automatiser-sa-reduction-de-donnees-spectro-sans-coder-avec-arp-lite/

Extension et ouverture

Ce processus est un exemple. La chaîne de traitement décrite ici a pour objectif de présenter une entrée en matière en donnant les clés pour démarrer et faire son propre processus. Lors de traitements automatiques, il est important notamment de tenir compte de toutes les situations qui peuvent survenir en fonction de leurs probables fréquences d'apparition (coupure de courant en plein milieu du traitement, données incomplètes, etc..).

Il en est de même pour la solution choisie ici. L'avantage de cette solution, n8n, est de pouvoir répondre à des configurations et des niveaux de connaissances techniques multiples. Un utilisateur avancé pourra par exemple utiliser le bloc ***Execute command*** (le fameux !) pour lancer un script Python dans lequel il aura intégré des traitements puissants tout en gardant la simplicité graphique pour la gestion de la chaîne globale. Il n'est pas impossible qu'un nouvel article voie le jour en utilisant cette possibilité prochainement.

D'autres solutions existent en fonction des usages :

- La solution NodeRed est une alternative très proche à celle utilisée ici, également en JavaScript. La documentation est très bien faite et il existe beaucoup de vidéos et articles avec des exemples pertinents en ligne.

https://nodered.org/

- On notera aussi la solution Airflow en Python. Cette dernière est plutôt destinée à des développeurs Python. Elle nécessite une configuration matérielle plus importante et un serveur Linux, ou encore Docker. Cela dit, elle permet de répondre à des besoins plus importants en termes de traitement. C'est par exemple la solution qui est mise en place pour l'association 2SPOT (https://2spot.org/FR/) qui bénéficie de nombreux ciels clairs au Chili et donc d'un volume de données à traiter régulier et conséquent.

https://airflow.apache.org/

En espérant que cet article puisse être utile à vos usages, n'hésitez pas à me contacter si vous avez des questions ou si vous souhaitez échanger sur ces solutions bien sûr.

Merci de m'avoir lu jusqu'ici.