A||GO Documentation

Sebastien Campion

Jan 09, 2019

Contents

Allgo is a Scientific Software As A Service (*SAAS*) platform.

CHAPTER 1

Introduction

Users can **upload** their data after choosing the software they wanted to test. Once the data is processed, the user can **download the results**.

For instance, Samusa is an algorithm for detecting music and speech in an audio file. Upload an *MP3* file and obtain a text file containing all the timestamps of each kind of segment.

A web interface and *HTTP REST API v1* are provided in order to test and integrate AllGo with your development.

From a provisioning point of view, several methods are available (see *Deploy an application*) to make your software deployement as easy as possible.

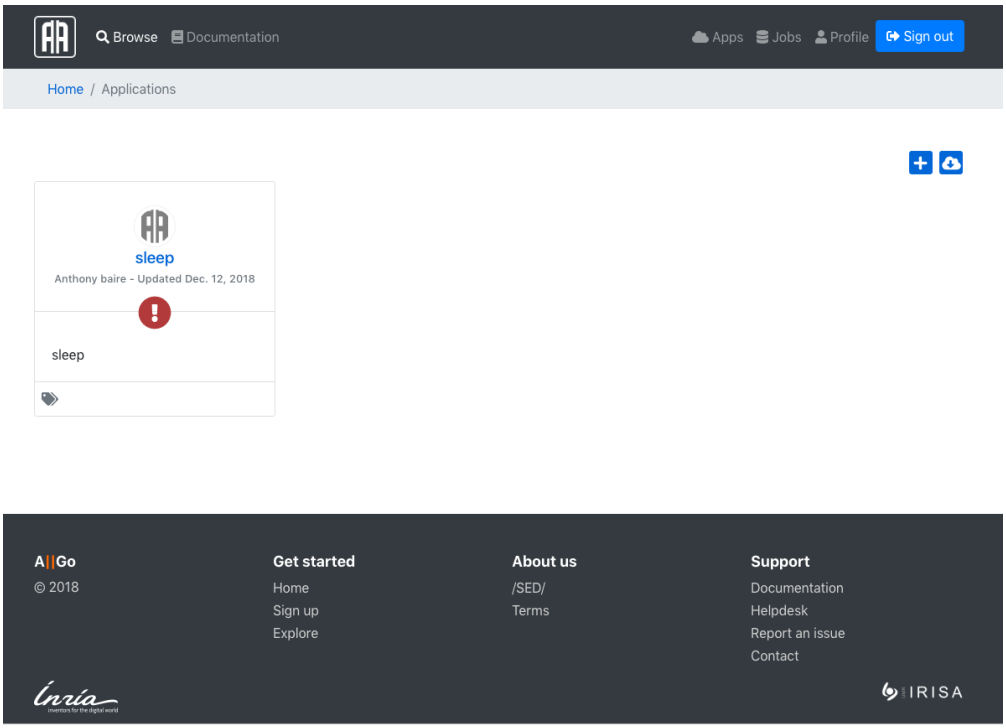
Contact : allgo@inria.fr

CHAPTER 2

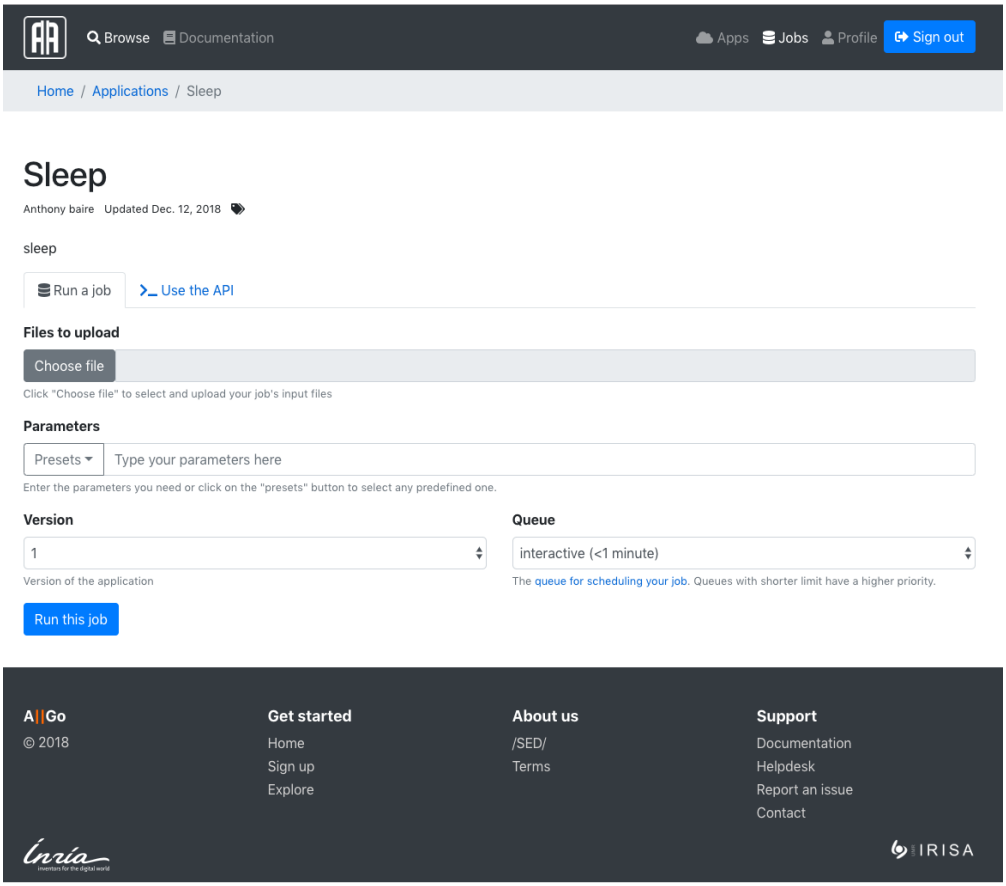
User's guide

Prerequisites : create an account and validate it by email.

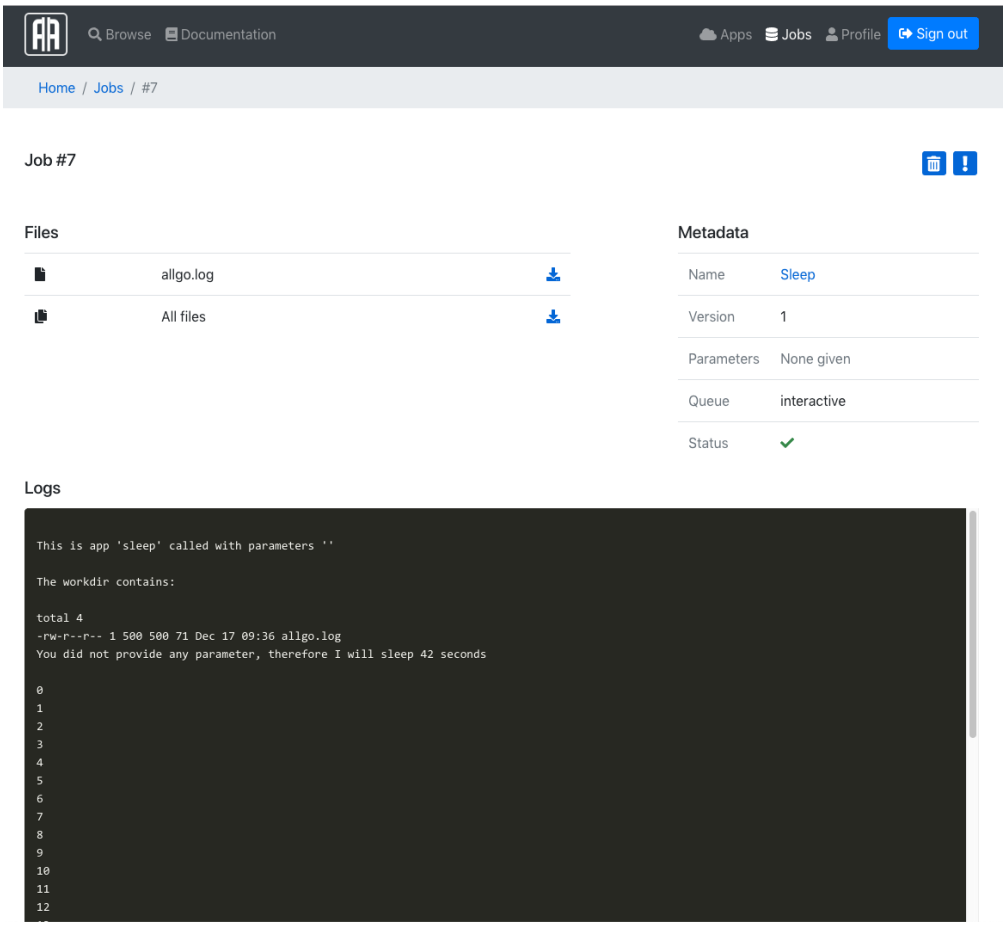
1. Choose your application



2. Upload your files (Upload



3 Download your results


Browse
Documentation
Apps
Jobs
Profile
Sign out
Home / Jobs / #7
Job #7
Files
allgo.log
All files
Metadata
Name
Sleep
Version
1
Parameters
None given
Queue
interactive
Status
Logs
This is app 'sleep' called with parameters ''
The workdir contains:
total 4
-rw-r--r-- 1 500 500 71 Dec 17 09:36 allgo.log
You did not provide any parameter, therefore I will sleep 42 seconds
0
1
2
3
4
5
6
7
8
9
10
11
12

CHAPTER 3

Table of contents

3.1 HTTP REST API v1

In order to use the AllGO HTTP API, you have to copy your **private token** from your profile [page](https://allgo.inria.fr/users/edit)

3.1.1 Post a job

POST /api/v1/jobs
Submit a new job

Form Parameters

- **job[webapp_id]** – int webapp identifier
- **job[param]** – a string used as parameter
- **files[0]** – a first file
- **files[1]** – a second file, etc ...

Request Headers

- Authorization – Token token=<your_private_token>
- Content-Type – multipart/form-data

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id":40155,
  "url":"https://allgo.inria.fr/api/v1/jobs/40155"
}
```

Here is a example with curl

```
curl -H 'Authorization: Token token=<your_private_token>' -X POST -F job[webapp_id]=
↪<service_id>
    -F job[param]='' -F files[0]=@<path_to_file0> -F files[1]=@<path_to_file1>
    https://allgo.inria.fr/api/v1/jobs
```

This request will return you, either the errors, or if it's ok, your **job_id**

3.1.2 Get results

Third, check the resulting job to get the url files with :

GET /api/v1/jobs/(int: job_id)
Retrieve job informations

Request Headers

- Authorization – Token token=<your_private_token>

Example response:

```
{
 "40155":
   {"conv1_samusa.txt":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/conv1_samusa.txt",
    "conv1.json":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/conv1.json",
    "conv1.mp3":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/conv1.mp3",
    "allgo.log":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/allgo.log"
    },
 "status":"done"
 }
```

Each file could be downloaded with the link associated. http

```
GET /api/v1/jobs/40155 HTTP/1.1
Host: allgo.inria.fr
Accept: application/json
Authorization: Token token=<your_private_token>
```

curl

```
curl -i https://allgo.inria.fr/api/v1/jobs/40155 -H 'Accept: application/json' -H
↪'Authorization: Token token=<your_private_token>'
```

wget

```
wget -S -O- https://allgo.inria.fr/api/v1/jobs/40155 --header='Accept: application/
↪json' --header='Authorization: Token token=<your_private_token>'
```

httpie

```
http https://allgo.inria.fr/api/v1/jobs/40155 Accept:application/json Authorization:
↪'Token token=<your_private_token>'
```

python-requests

```
requests.get('https://allgo.inria.fr/api/v1/jobs/40155', headers={
    'Accept': 'application/json',
    'Authorization': 'Token token=<your_private_token>',
})
```

response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "40155":
   {"conv1_samusa.txt":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/conv1_samusa.txt",
    "conv1.json":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/conv1.json",
    "conv1.mp3":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/conv1.mp3",
    "allgo.log":"https://allgo.inria.fr/datastore/6/1/
↪0fe2bc68b835e9a1f681e38d5e87001ef955e345/allgo.log"
    },
 "status":"done"
 }
```

3.2 Deploy an application

This is a brief and simple tutorial about how you can deploy your app/algorithm on A||go. We try to make it really simple, so, if something looks complicated, send us an email allgo@inria.fr

3 steps are necessary to deploy an app :

1. Declare your application on the website
2. Connect by ssh to install your soft
3. Setup your *entrypoint*

Note: Note that the required knowledge you need to have, is only how to install your app. We remind you that A||go only works for application without UI, running on linux, without workflow. You need to add your public SSH key in your profile, so you'll be able to access to your machine later.

3.2.1 1. Create an application on the website

In order to create an new app, you need to use the website by filling a form. You need to specify the application name, the email contact if it's not yourself, a description (in the form of a README) and select if your application should be public or private.

Note: If you need you can access the "Advanced" tab where you can select a different operating system (the default is Debian 9), the memory limit and default queue type.

After creating your app, you'll be redirected to a page where you'll be ask to perform the setup of the sandbox.

3.2.2 2. SSH setup

An ssh command allow you to connect and setup your software.

You are free to install any package you need (compilers, JRE, python, etc, . . .) and of course your software.

The last step is the entrypoint a file (usually a binary) that will be call to execute your application.

3.2.3 3. Entrypoint

The entrypoint is the file that is launched by allgo to process a job.

The user creating the job may provide command line arguments and input files. The arguments are passed to the entrypoint command, and the input files are located in the current working directory.

For example, an app that works on text files could have the following entrypoint:

```
#!/bin/sh
for file in *.txt;
do
  echo "Processing $file"
  /usr/local/bin/myapp --input "$file" --output "$file.output" "$@"
done
```

In this example, myapp is run on each input text file and store the result as FILENAME.output. The extra argument "$@" is there to forward the command line arguments provided to the entrypoint.

Important: Once, you installed your application, do not forget to **commit** your work through the website (button commit) or it will be lost.

3.2.4 Testing the application

After installing your app, come back on the public application page and create a job with the necessary file(s) and parameters. It's eventually possible to change the *Queue type* queue type by accessing the "advanced" tab.

If your job doesn't work, you'll need to fix it by connecting to your sandbox using SSH or by editing your Dockerfile and commit your work.

3.2.5 Update your application

Describe the parameters that can be updated and tags.

- Icon
- tags
- description
- default queue

3.3 Run a Job

3.3.1 Parameters

Application often needs parameters. These one can be forwarded by a string to the application. In order to facilitate their usage, some presets can be defined by the author . If you select one of them in the web form, the entry field will be automatically filled. Otherwise, the application documentation should help.

3.3.2 Queue type

A job will be executed in a given queue depending of the time processing. The system has 3 queue types:

- Interactive: less than a minute
- Standard: less than 20 minutes
- batch: less than a day

Warning: If a job lasts more than the queue max duration, it will time out and be aborted.

3.3.3 Quotas

User can be limited by quotas. For exemple, the samusa audio segmentation tools is limit to 50 MB of audio files. If you need more, please contact the author of the application.

3.4 Notebook integration

We will explain in this document how to use AllGO with a python notebook.

An example is available here : https://gitlab.inria.fr/allgo/notebooks/ndsafir

3.4.1 1. Install allgo module

A client library is available at https://pypi.org/project/allgo/

Install it with :

```
pip install allgo
```

3.4.2 2. Create an application

```
app = allgo.App('ndsafir', token="ead123baaef55412")
```

Note: token in optional, if you already provide your token with an env variable ALLGO_TOKEN or create a file ~/.allgo_token (without breakline)

3.4.3 3. Submit a job :

```
files = {'files[0]': open('tmp.png', 'rb')}
params = '-nopeaks 1 -2dt false -noise 0 -p 1 -bits 8 -iter 5 -adapt 0'
app.run(files=files, params=params)
```

In this example, we send one file 'tmp.png' and a string of parameters 'params'. Run is blocking, when finish all files produce by AllGo are downloaded in the current directory. *ndsafir* application create a file *output.png* in that case, so display the result with :

```
import matplotlib.pyplot as plt
img = mpimg.imread('output.png')
plt.imshow(img)
```

3.5 Complex GUI

You will discover in this example more complex GUI with IPyWidgets

Illustration du traitement en mode interactif

En utilisant les curseurs (*slider*) ci-dessous, vous pouvez modifier le choix des 2 paramètres cles de ND-Safir, et visualiser les résultat (en cliquant sur "Run interact").

```
def wrapper_ndsafir_for_interact(nbIter,patch):
    ndsafir.ndsafir_for_interact(TOKEN, APP_ID, nbIter, patch)

#wrapper_ndsafir_for_interact()

from ipywidgets import interact, interactive, fixed, interact_manual
interact_manual(wrapper_ndsafir_for_interact, nbIter=(0,15),patch=(1,4));
```

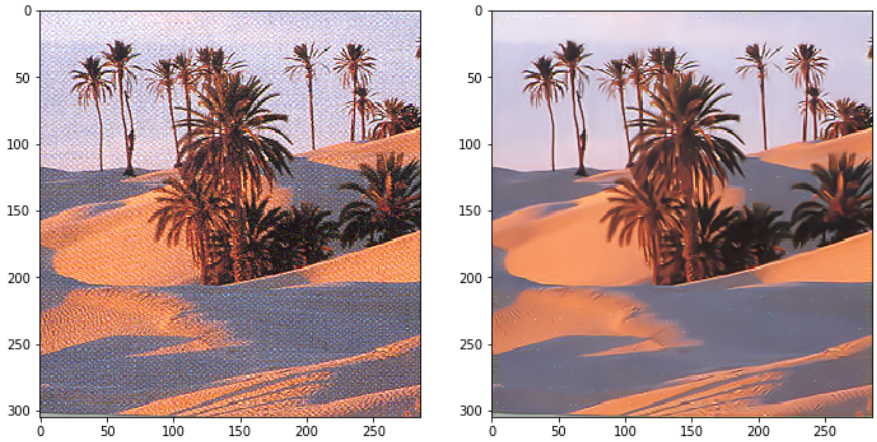
nbIter 7

patch 2

Run Interact

```
AllGo job_id : 16821
Waiting job end ..
Status : done

<matplotlib.figure.Figure at 0x7f4c23fe67f0>
```



3.6 Frequently Asked Questions

3.6.1 I found an issue or have a request, how can I report it ?

If you found a bug or any issue, you can report it on the Allgo repository. For support requests, you may contact us in private on INRIA Helpdesk or by email.

3.6.2 How long my files stay in the plateform ?

Job older than **one month** are deleted.

3.6.3 What about my data ?

Your data cannot be used outside the institute. They may be used only for debug of the application by the author.

3.6.4 What about the roadmap ?

This new version A||GO18 is almost isofunctional except for documentation and demonstrators because we will integrate notebooks with Jupyter into the platform. Indeed, notebooks are a most approriate tool for this kind of usage, you will have better GUI widgets, portability and reliability.

Notebook are planned to be integrated in Q1 2019.

We also looks in the public issue tracker to schedule our developpements. The job progress feature will probably be included in this release also.

3.7 Glossary

MP3 MPEG-1/2 Audio Layer III

SAAS Software As A Service

HTTP Routing Table

/api

```
GET /api/v1/jobs/(int:  job_id), 10
POST /api/v1/jobs, 9
```

Index