Getting Started with HelioCloud
v1.0, Antunes, Mar 14 2022

The HelioCloud Project (heliocloud.org) is replicable cloud computing and big data environment and tool set for the Heliophysics research community.

First you need to get access to GSFC's cloud environment, which uses Amazon Web Services and is called SMCE. Within SMCE, the specific environment we are using is called HelioCloud (Heliocloud.org) and primarily uses Jupyter Notebooks and Daskhub as the development environment.

Once you have access, we'll walk you through how to get into the actual cloud, set up your environment, and

****

1) Get set up with SMCE HelioCloud access as a 'DaskHub user'
(not power user)

Request access to SMCE and get the required documents by emailing Brian Thomas (brian.a.thomas@nasa.gov) with the following message:

Hello Brian,

I would like to be given access to SMCE to use HelioCloud as part of the APL HelioCloud development/test effort. Please send me the required forms and instructions.

My project is [list mission, grant, institute, etc here to briefly justify].

<signed here>

Read the 3 SMCE documents, sign the last PDF and email it to:
Jay Ellis (n.jay.ellis@nasa.gov)
CC: aaron.skolnik@nasa.gov, brian.a.thomas@nasa.gov

After SMCE access is granted, email Brian Thomas your request to be added to the HelioCloud Daskhub as a separate email, using language similar to the above.

(Other resources for power users include GitLab and the AWS Console, not needed in this case). Include your name including middle initial, and phone number.

Please track how long this process took and any issues via the Feedback form.

****

2) OPTIONAL-- Set up Multi-Factor Authentication (MFA)-- only required if you need a Console account, i.e. power user!

Generally this is for people who intend to create datasets for placing onto S3 or otherwise needing additional functionality outside the scope of the Daskhub/Notebook environment.

You will receive from Brian a UserId & a temporary password.

Password requirements are 12 characters, one upper, one lower, one 'special'

You will be given a signin URL, probably:
Sign-in URL: https://smce-helio.signin.aws.amazon.com/console
User name: *********

* Log in, change password. Rule: need to have at least 12 characters in the password, one upper case, one lower case, one ‘special’ (think !@#$% characters) and one number.
* DON'T LOG OUT YET, you need to setting up MFA:

* First time login, go to “My Security Credentials” tab (NE corner, pull down from your user name which appears next to the notification ‘bell’)

* In the security credentials tab, set up “Multi-factor Authentication” (MFA), click that to add a device.

* Choose “Use QR Code”. You will need to have added the “Google Authenticator” app (or similar) to your phone. In that app, you can click the “+” at the SE bottom corner and it will scan the QR code.

* Then add the requested 2 examples and log out.
* Then log back in with MFA. You should be set then

Please track how long this process took and any issues via the Feedback form.

****

3) Set up your personal environment.

The link is:
https://daskhub.helio.mysmce.com/

You will be prompted with 'Server Options', just keep the default of 'small server' and click 'Start'

This launches a virtual machine in the heliocloud, and puts you into the JupyterHub notebook environment, your primary file manager and editor.

More setup! In the file manager, go into the 'efs' directory and make a home directory with first_initial last_initial last_name, like the others. This is your primary shareable work area.

Next, enter the 'Tutorials' area to see what support exists.

OPTIONAL: As an optional step, you can (now or later) customize your Python environment with additional packages you need. There is 'Conda_instructions_for_cloud.ipynb in the efs/Tutorial directory'. Click on that, read it, set up an environment using its commands. IF YOU DO NOT DO THIS, THAT'S OKAY. You can do it later, and update it later.

Note: if the step of 'open a terminal' does nothing in your Jupyter session, go to the 'Help' menu and select 'Launch Classic Notebook', and in that you can use the 'New' dropdown and start a terminal (gets around a browser bug).

Why to do this optional step? Your 'conda' environment will always contain the packages and software you tell it to import. This is how you can vary from the default generic account to add tools you need.

Please track how long this process took and any issues via the Feedback form.

****

4) Start coding

Either FILE->NEW->NOTEBOOK or the more visual FILE->NEW LAUNCHER. Choose a Python kernel from the Notebooks offered.

Now you can code. Basics on using a Notebook:

* items in a cell are either code ('code') or text ('markdown' or 'raw'). Code gets run, text is there to read and document as you go.
* hitting return in a cell lets you add more lines. Clicking on '+' opens a new cell. Try to make each cell be a code block.
* To run code, click on the cell where you want the program to start and click on the 'run' triangle. That runs that entire cell but nothing else. Each time you click on 'run', it runs the cell's code then advances to the next cell.

You can also use the 'Run' menu. Let's try some sample code. Below, create some cells to contain code. Cells are ORGANIZATIONAL units, not COMPUTATIONAL units, which is to say there are there for you and to make it easier to walk through your code step by step, they do not have any connection to Python or how the code will run. You can put everything in 1 cell and it will work, but using cells is a better way to develop code in Notebooks. Each cell is labeled by the Notebook with [#], as the sample below.

[1]:
```
print("Hello World")
```
[2]:
```
import hapiclient, hapiplot
```

Save your code (because although Notebooks auto-save, you should always save your code periodically) by clicking on the disk icon or using FILE->SAVE NOTEBOOK AS. Also, if you need to rename it, because it defaulted to 'Untitled.ipynb', either right-click on the tabbed name and choose 'Rename Notebook', or FILE->RENAME NOTEBOOK.

Try running cell 1, it works. Oh no, cell 2 fails because hapiclient isn't installed! Let's fix that.

[1]:
```
print("Hello World"()
```
[2]:
```
!pip install hapiclient
!pip install hapiplot
```
[3]:
```
import hapiclient, hapiplot
```

That worked, now let's add some code:

[4]:
```
server     = 'https://cdaweb.gsfc.nasa.gov/hapi'
dataset    = 'WI_H0_MFI@0'
parameters = 'BF1,BF1LOG'
start      = '2020-11-12T00:00:30Z'
stop       = '2020-11-14T00:00:30.000Z'

data, meta = hapiclient.hapi(server, dataset, parameters, start, stop)
```

[5]:
```
hapiplot.hapiplot(data,meta)
```

[6]
```
print(meta)
print(data)
```

Wow, looks like we have real data, in a pandas array of some sort. Life is good.

Please track how long this process took and any issues via the Feedback form.

****

5) More advanced cloud usage

Above, we've replicating using a Jupyter Notebook to do the same things you would do on your local machine, only now you're operating in a VM and using a Notebook. The power of cloud comes in either being able to run a program on multiple nodes (Dask), or accessing big data sets stored in the cloud (S3), or both. Try these 3 demos to get a feel for things. They are in /efs/Tutorials.

* Dask-Gateway-Example.ipynb
* S3_cdf_demo.ipynb
* S3_Dask_demo.ipynb

Please track how long this process took and any issues via the Feedback form.

****

6) (OPTIONAL) Creating Access Keys for Writing to S3 (advanced)

S3 bulk storage is open to everyone as read-only. If you need to write to S3, you will need to get access via the terminal. In the terminal:

i) Copy the mfa-aws script to your home directory (*cp /efs/bathomas/mfa-aws ~*)
ii) Create a keys file using your AWS cert from the optional step 2
iii) Create a credentials file that looks like this: (*mkdir .aws; nano .aws/credentials*)

```
[smce]
aws_access_key_id = PASTE FROM AWS SECURITY PAGE
aws_secret_access_key = PASTE FROM AWS SECURITY PAGE
region = us-east-1
```

output = json

iv) Run the script to authenticate (*~/aws-mfa smce*)
v) Copy-and-paste the "export AWS_SESSION_TOKEN=lotsofcharacters" into the terminal

It will complete authentication and indicate how long those keys are valid. Every time you need to re-authenticate, just repeat steps iv & v.

****

7) Tackling MMS

We have MMS data in S3. To tackle this is the next step.

[tbd]