IOOS Cloud Sandbox Setup and User Guide

Patrick Tripp

RPS Group

March 23, 2020

[IOOS Cloud Sandbox Setup and User Guide 2](#_Toc35867505)

[Base Platform Setup 2](#_Toc35867506)

[Environment Setup 3](#_Toc35867507)

[Enable Passwordless SSH 3](#_Toc35867508)

[User Setup 3](#_Toc35867509)

[Create the AMI 4](#_Toc35867510)

[What it does: nosofs-setup-instance.sh 4](#_Toc35867511)

[NOSOFS Model Setup 4](#_Toc35867512)

[Create an EC2 instance to use as the main interactive machine 5](#_Toc35867513)

[Obtain and Build the Source Code 5](#_Toc35867514)

[Obtain the fixed field files 6](#_Toc35867515)

[Obtain the initial conditions and forcing data 6](#_Toc35867516)

[Setup and Run the Forecast 6](#_Toc35867517)

[LiveOcean Model Setup 7](#_Toc35867518)

[Obtain the model and related scripts 7](#_Toc35867519)

[Build the model 7](#_Toc35867520)

[Obtain the fixed field files 7](#_Toc35867521)

[Obtain the initial conditions and forcing data 7](#_Toc35867522)

[Setup and Run the Forecast 8](#_Toc35867523)

[Reproducible Tasks and Workflows 8](#_Toc35867524)

[Directory structure 8](#_Toc35867525)

[Setup the machine configuration files for the forecast and/or post processing 8](#_Toc35867526)

[Setup the job configuration files 8](#_Toc35867527)

[Run the workflow 8](#_Toc35867528)

[Machine Config Files 9](#_Toc35867529)

[Job Config Files 10](#_Toc35867530)

[Plotting 10](#_Toc35867531)

[Easy Viewing 10](#_Toc35867532)

[Re-packaging After Changes 11](#_Toc35867533)

[Creating Custom Workflows 11](#_Toc35867534)

[Prefect Flows 11](#_Toc35867535)

[Prefect Tasks 12](#_Toc35867536)

[Appendix A – Python Workflows UML Class Diagram 13](#_Toc35867537)

[Appendix B – Models Tested 14](#_Toc35867538)

[ROMS Models Tested 14](#_Toc35867539)

[Untested ROMS Models 14](#_Toc35867540)

[FVCOM Models Tested 14](#_Toc35867541)

[Appendix C – Useful Links 16](#_Toc35867542)

[Appendix D – Notes 16](#_Toc35867543)

IOOS Cloud Sandbox Setup and User Guide

#### Base Platform Setup

Cloud platform: Amazon Web Services

Requires: An AWS account with an available Virtual Private Cloud and subnet with needed routes, gateways, roles, and security groups setup. See the AWS documentation on how to set these up.

Overview: This section walks through the creation of a base image that will be re-used each time a new machine is created in the cloud.

(TODO: Create a script to do this using the AWS CLI or Boto3)

Using the AWS Console reate a c5n.18xlarge instance with EFA enabled network:

Open the EC2 console

Click “Launch Instance”

Select the “CentOS 7 (x86\_64) - with Updates HVM” AMI in AWS Marketplace

Click “Continue”

Select the “c5n.18xlarge” Instance Type

Click “Next: Configure Instance Details”

Change the following options:

Network: Select or create a VPC

Subnet: Select or create a subnet

Auto-assign Public IP: Enable

Placement group: Check “Add instance to placement group”

Placement group name: Add to existing or create new placement group.

Elastic Fabric Adaptor: Enable

IAM role: Select or Create new IAM role that has full EC2 and S3 access.

CPU options: Check “Specify CPU options”

Threads per core: 1

(FVCOM does not run with Hyperthreads, and they provide no speed benefit in these models.)

Shutdown behavior: Stop

File systems: “Add file system” if one is already created, or “Create new file system”

Note: EFS requires a security group that will be used to allow the EC2 instances access to it.

Make sure the “User data:” section under “Advanced Details” is populated with #clound-config data. If not, re-select the “File systems” and it should populate.

Click “Next: Add Storage”

Check “Delete on Termination”

Click “Next: Add Tags”

Add tags to help identify and track the usage of this instance.

e.g. Key: Project, Value: IOOS-cloud-sandbox

e.g. Key: Name, Value: Setup base image

Click “Next: Configure Security Group”

Create a new or Select an existing security group

Make sure that SSH access is allowed from Source: “My IP” and any other IPs where access is needed. (I discourage an open policy of “Anywhere”).

Also add the security groups for both EFS and EFA (refer to the AWS documentation for setting these up.)

Click “Review and Launch”

Double check the settings and click “Launch”

Choose or create a key pair to use and check the “I acknowledge …” box.

Click “Launch Instances”

Click “View Instances” (bottom right corner) this will take you to the EC2 Console

Select the new instance and copy the Public DNS or Public IP.

SSH into the newly created instance and proceed to Environment Setup below.

From a terminal:

ssh -i <path/your\_key.pem\_specified\_above> centos@<the\_public\_DNS\_or\_IP\_copied\_above>

(Click the “Connect” button at the top of the EC2 Console page for instructions if you need help connecting to the instance in other ways.)

##### Environment Setup

sudo yum update

sudo yum install git

git clone https://github.com/asascience/Cloud-Sandbox.git

cd Cloud-Sandbox/scripts

./nosofs-setup-instance.sh

Running the above will install all of the required libraries, drivers, and software dependencies for the sandbox.

##### Enable Passwordless SSH

ssh-keygen -t rsa

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

(Although a password protected key is recommended, it is inconvenient. The key will need to be added to the ssh-agent, with the password entered, everytime the system reboots.)

Add something similar to the following to /etc/ssh/ssh\_config. This will prevent mpirun from failing when trying to connect to new machines in the subnet. The below Host should match your subnet.

Host ip-10-0-0-\*

CheckHostIP no

StrictHostKeyChecking no

Host 10.0.0.\*

CheckHostIP no

StrictHostKeyChecking no

##### User Setup

This is the time to setup additional users if needed. If done later, a new AMI image will need to be created so that the userid and their SSH keys will be valid each time a new machine is created.

Add the user and optionally add them to additional groups.

Create SSH keys for the user and add the public key to their .ssh/authorized\_keys file as in the previous step.

##### Create the AMI

Via the AWS Console:

Stop the instance (DO NOT TERMINATE IT YET)

Actions 🡪 Instance State 🡪 Stop

Create a new AMI image from the instance. This is the base image that will be used for all subsequent work.

Actions 🡪 Image 🡪 Create Image

Enter a name and description, use the default settings for everything else. Delete on Termination should be checked, otherwise the volume needs to be manually deleted everytime a new machine is terminated.

Terminate the instance.

##### What it does: nosofs-setup-instance.sh

The nosofs-setup-instance.sh script above performs the following tasks:

Updates the linux operating system via yum update.

Downloads and installs the compilers (GCC 6.5).

Installs the required software and drivers:

Environmental modules support  
Python3  
AWS Boto3  
AWS EFA drivers  
IntelMPI 2019+

ffmpeg

python3 module dependencies

Installs the required libraries:

HDF5  
NetCDF4  
GRIB2  
PNG  
Jasper  
NCEPLIBS (required for NOSOFS prep)  
produtil (required for NOSOFS scripts)

Creates environment modules for the above libraries.

#### NOSOFS Model Setup

Now that there is a base operating system image that contains all of the prerequisite drivers and software, the models and related experiments can be setup.

##### Create an EC2 instance to use as the main interactive machine

Using the AWS Console:

Navigate to the EC2 Dashboard

Click Launch Instance

Select “My AMIs” and Select the AMI created in the previous section.

This instance only needs to be large enough to support builds and other tasks in an interactive terminal,

t3.small is sufficient. A larger instance may be needed if it needs to support multiple concurrent users.

Select t3.small and click “Next: Configure Instance Details”

Change the following options:

Network: Choose the VPC used in the previous section.

Subnet. Choose the subnet used in the previous section.

Auto-assign Public IP: Enable

IAM role: Select or Create new IAM role that has full EC2 and S3 access.

Click “Next: Add Storage”

Click “Next: Add Tags” and add tags to help identify resources and track usage.

Click Next: Configure Security Group

Select “Select an existing security group” and select the security groups for SSH access and EFS that were created earlier.

Click Review and Launch

Review the settings and click “Launch”

Choose or create a new keypair to use to connect to this instance.

Click “Launch Instances”

After a minute or so, log in to the instance via ssh.

The Public DNS and IP address can be found in the Intances pane of the EC2 Dashboard.

From a terminal:

ssh -i <path/your\_key.pem\_specified\_above> centos@<the\_public\_DNS\_or\_IP>

(Click the “Connect” button at the top of the EC2 Console page for instructions if you need help connecting to the instance in other ways.)

##### Obtain and Build the Source Code

The model needs to be built on the shared EFS/NFS mounted filesystem so it is available to all of the nodes that will be provisioned.

Use the build scripts to build the models and optional prep step components.

cd /save

git clone https://github.com/asascience/nosofs-NCO

or if you have ssh keys enabled with github:

git clone git@github.com:asascience/nosofs-NCO.git

cd nosofs-NCO/sorc

./ROMS\_COMPILE.sh # ROMS prep and models. Edit it for full or partial builds.

./FVCOM\_COMPILE.sh. # FVCOM prep models. Edit it for full or partial builds.

##### Obtain the fixed field files

These files are too large to easily store on github and need to be obtained elsewhere.

You can run the below script to download all of the fixed field files from the IOOS-cloud-sandbox S3 bucket.

Edit the script to only download a subset. The following assumes the Cloud-Sandbox repository was cloned in the home directory.

mkdir -p /save/nosofs-NCO/fix

cd /save/nosofs-NCO/fix

~/Cloud-Sandbox/scripts/getFixedS3.sh

Alternatives:

Manually download from the ioos-cloud-sandbox S3 bucket.  
Example: wget <https://ioos-cloud-sandbox.s3.amazonaws.com/public/nosofs/fix/cbofs.v3.2.1.fix.tgz>

Download from NOAA webserver (requires the shared/ and the region specific folder)  
Example: wget <https://www.nco.ncep.noaa.gov/pmb/codes/nwprod/nosofs.v3.2.1/fix/cbofs>  
Note: Some of the fix files from NOAA/pmb above contain errors that prevent the model from running.

##### Obtain the initial conditions and forcing data

If using the Cloud-Sandbox worflows described later, the forcing data can be automatically downloaded if it is still available on NOMADS.

Alternatives:

1. Manually download current operational data from NOAA NOMADS server. Operational NOSOFS products are retained for two days. <https://nomads.ncep.noaa.gov/pub/data/nccf/com/nos/prod/>
2. Run the prep step for the desired forecast cycle:

Download the meteorological forecast data from NOMADS.

<https://nomads.ncep.noaa.gov/pub/data/nccf/com/nam/prod/>  
<https://nomads.ncep.noaa.gov/pub/data/nccf/com/rtofs/prod/>

<https://nomads.ncep.noaa.gov/pub/data/nccf/com/etss/prod/>

Download any required restart files from NOMADS.   
Example: *(Note: The restart file needs to be renamed, details are in the getICsROMS.sh, getICsFVCOM.sh script.)* <https://nomads.ncep.noaa.gov/pub/data/nccf/com/nos/prod/cbofs.20200108/nos.cbofs.init.nowcast.20200108.t06z.nc>

Run the prep job using the jobs/runprep.sh script.

1. Download test case ICs and forcing data from S3.  
   Example: <https://ioos-cloud-sandbox.s3.amazonaws.com/public/cbofs/ICs.cbofs.2019100100.tgz>
2. Provide some other forcing data (archived, experimental, etc.)

##### Setup and Run the Forecast

See the section “***Reproducible Tasks and Workflows”***.

#### LiveOcean Model Setup

If not done in the previous section, create an EC2 instance to use as the main interactive machine.

##### Obtain the model and related scripts

Recommended path: /save/LiveOcean

Options:

1. Clone from RPS/ASA GitHub repository and run the scripts to obtain other required files. Access permissions must be provided by UW to access the LO\_ROMS repository.

cd /save

git clone <https://github.com/asascience/LiveOcean.git>

cd LiveOcean  
./get\_LO\_ROMS.sh

1. Download directly from UW (outside the scope of this documentation. See LiveOcean/NOTES\_Instructions for more details.)

##### Build the model

./LO\_COMPILE.sh

##### Obtain the fixed field files

Path: /save/LiveOcean/LiveOcean\_data/grids

Options:

1. The cas6 grid files are included in the RPS/ASA LiveOcean git repository and need to be untarred. The following script will retrieve the data from AWS S3 and untar it.

./get\_LiveOcean\_data.sh

1. Download from University of Washington.

##### Obtain the initial conditions and forcing data

If using the Cloud-Sandbox worflows described later, the forcing data can be automatically downloaded if it is still available on UW servers and if access has been provided by UW.

Path conventions:

Forcing data example: /com/liveocean/forcing/f2019.11.06

Init/restart file example: /com/liveocean/f2019.11.05/ocean\_his\_0025.nc

Alternatives:

1. Download current operational data from UW server. *(Requires a user account)*
2. Download test case ICs and forcing data from S3 and untar.

https://ioos-cloud-sandbox.s3.amazonaws.com/LiveOcean/forcing/liveocean.forcing.f2019.11.06.tgz

https://ioos-cloud-sandbox.s3.amazonaws.com/LiveOcean/init/liveocean.init.f2019.11.05.tgz

##### Setup and Run the Forecast

See the next section “***Reproducible Tasks and Workflows”***.

#### Reproducible Tasks and Workflows

Github repository: https://github.com/asascience/Cloud-Sandbox

##### Directory structure

.

├── workflow # Python3 workflow sources

│ ├── cluster # Cluster abstract base class and implementations

│ ├── configs # cluster configuration files (JSON)

│ ├── job # Job abstract base class and implementations

│ │ └── templates # Ocean model input namelist templates

│ ├── jobs # job configuration files (JSON)

│ ├── plotting # plotting and mp4 routines

│ ├── services # Cloud agnostic interfaces and implementations e.g. S3

│ ├── tests # Misc. testing. (TODO: add unit testing)

│ ├── utils # Various utility functions, e.g. getTiling(totalCores), ndate(), etc.

│ └── workflows # Workflows and workflow tasks

├── scripts # BASH scripts for various tasks

└── README.md

##### Setup the machine configuration files for the forecast and/or post processing

cd Cloud-Sandbox/workflow/workflows

There are two cluster configuration files that need to be specified. These should be changed for your particular configuration. These correspond to the machine configuration used for the forecast and post processing flows.

Examples:

See: workflow\_main.py - This is the main entry point.

fcstconf = f'{curdir}/../configs/cbofs.config'

postconf = f'{curdir}/../configs/post.config'

##### Setup the job configuration files

These are provided as command line arguments to workflow\_main.py or a copy of it.

Examples:

jobs/cbofs.00z.fcst (forecast)

jobs/cbofs.plots (post)

##### Run the workflow

To run the workflow (and to optionally log to an output file and run as a background process):

Example:

./workflow\_main.py jobs/ cbofs.00z.fcst jobs/cbofs.plots 2>&1 cbofs.out &

Note: job2 is optional and will only run if job1 finishes without error.

If using the cloud, cloud resources will be provisioned for you, based on the mCHINE configuration files specified above. The workflows can also be run locally, using the “Local” provider. The local provider is specified in the cluster configuration file by setting "platform" to “Local”.

Example: configs/local.config:

{

"platform" : "Local",

"nodeCount" : 1,

"PPN" : 2

}

The default output directory for NOSOFS is /ptmp while the forecast job is running. When the job is complete, results are copied to /com.

*(TODO: Re-test Local and update if needed.)*

The cloud resources will be automatically terminated when each job finishes.

To customize the flows see: flows.py

To add any additional tasks, see: job\_tasks.py and tasks.py in the workflows folder.

To add additional Job class functionality or define new Job types, see the classes in the ./job folder.

To add additional Cluster class functionality or define new Cluster implementations, see the classes in the ./cluster folder.

See the ./plotting folder for plotting routines.

##### Machine Config Files

Note: The last five settings below are specific to the account being used. These are the resource identifiers for the resources created earlier and can be found in the AWS Console or with the AWS CLI.

Example: configs/cbofs.config

{

"platform" : "AWS",

"region" : "us-east-1",

"nodeType" : "c5n.18xlarge",

"nodeCount" : 2,

"tags" : [

{ "Key": "Project", "Value": "IOOS-cloud-sandbox" },

{ "Key": "Name", "Value": "cbofs-fcst" } ],

"image\_id" : "ami-0xxxx",

"key\_name" : "your-pem-key",

"sg\_ids" : ["sg-0xxxx1","sg-0xxxx2","sg-0xxxx3"],

"subnet\_id" : "subnet-0xxxx",

"placement\_group" : "cloud-sandbox-cluster"

}

Description of variables:

platform - The cloud provider being used. Currently AWS or Local (runs on local machine).

region - The AWS region to create your resources in.

nodeType - EC2 instance type.

nodeCount - # of instances to provision.

tags - The tags to add to these resources, used for tracking usage.

image\_id - For AWS, the AMI id to use for each instance.

key\_name - The private key file to use to ssh into the instances.

sg\_id1-3 - The AWS security groups to use.

subnet\_id - The subnet id to launch in.

placement\_group - The cluster placement group.

##### Job Config Files

Example: jobs/cbofs0.00z.fcst

{

"JOBTYPE" : "romsforecast",

"OFS" : "cbofs",

"CDATE" : "today",

"HH" : "00",

"COMROT" : "/com/nos",

"TIME\_REF" : "20160101.0d0",

"BUCKET" : "ioos-cloud-sandbox",

"BCKTFLDR" : "/nos/cbofs/output",

"NTIMES" : "34560",

"ININAME" : "",

"OUTDIR" : "auto",

"OCEANIN" : "auto",

"OCNINTMPL" : "auto"

}

Description of variables:

JOBTYPE - current options are "[roms|fvcom]forecast" and "plotting"

OFS - name of the forecast. Current options are "cbofs", "dbofs", "liveocean"

CDATE - run date, format YYYYMMDD or "today" = today's date

HH - forecast cycle, e.g. 06 for 06z forecast cycle.

COMROT - common root path where output will be

TIME\_REF - reference time of the tidal forcing data being used

BUCKET - cloud storage bucket where output will be stored

BCKTFLDR - cloud storage folder, key prefix

NTIMES - number of timesteps to run this forecast

ININAME - The path/name of the INI/restart file to use.

OUTDIR - model output directory. "auto" = automatically set this, based on CDATE, etc.

OCEANIN - name of the ocean.in file to use. "auto" = use template.

OCNINTMPL - template file to use.

#### Plotting

##### Easy Viewing

To view the plots without having to copy them to the local machine, run the simple HTTP server that is included with Python.

Example:

cd /com/nos/plots/cbofs.20200310

python -m SimpleHTTPServer 8000

Terminal output:

Serving HTTP on 0.0.0.0 port 8000 ...

Port 8000 needs to be open to inbound traffic. Add this in the AWS EC2 Dashboard following the steps below.

<https://console.aws.amazon.com/ec2>

Select “Instances” in the left hand navigation bar.

Select the machine currently logged into.

In the “Desription” pane, to the right of “Security groups”, click the Security group that also allows SSH access to the instance.

Click the “Inbound” tab in the bottom pane.

Click “Edit”

Click “Add Rule” and set the following:

“Type”, select “Custom TCP Rule”

“Port Range”, enter 8000

“Source”, select “My IP”

Click “Save”

In a browser window, navigate to the instance public IP or DNS:8000

e.g. <http://ec2-3-235-2-212.compute-1.amazonaws.com:8000/>

Ctl-C will stop the server.

##### Re-packaging After Changes

If changes are made to the plotting module, it will need to be re-packaged before those changes are available to other nodes when running a workflow.

cd ../workflow

python3 ./setup.py sdist

This will place the created package in workflow/dist. It is automatically pushed to all hosts in the cluster when running the workflow.

Note: Occassionally, if the new instance is slow to boot up, mpirun or dask will give a connection timout or a connection refused error message. Retrying the workflow usually works fine. If this is a regular occurance, in cluster/AWSCluster.py there is a “sleep(60)” after the machine is created to give it time for the sshd daemon to start. Increase this sleep time a little and the problem should go away.

#### Creating Custom Workflows

##### Prefect Flows

There are currently two pre-defined Prefect flows in ./*workflow/workflows/flows.py,* fcst\_flow and plot\_flow.

Flows are a directed acyclical graphs. Each vertice is a task and each edge is a dependency. Tasks run in the order given by their dependencies and otherwise run anynchronously by the underlying Prefect framework. Each task is a callable function with the @task annotation. The tasks are defined in the three files, tasks.py, job\_tasks.py, and cluster\_tasks.py.

There are multiple ways to specify the dependencies (edges). Each predefined flow shows a different way of doing it. There are also different ways of creating a Flow. Refer to the Prefect documentation in Appendix C.

For example: fcstflow uses the .add\_edge function to specify the dependencies.

*fcstflow.add\_edge(cluster, fcstjob)* – states that the cluster task precedes the fcstjob task.

plotflow uses a different but equivalent way, it uses upstream\_tasks=[] to specify the dependency relationships.

Example:

*plotjob = tasks.job\_init(postmach, postjobfile, upstream\_tasks=[pmStarted])* – states that pmStarted precedes the plotjob task.

Another way demonstrated in plotflow is:

*pngtocloud = tasks.save\_to\_cloud(plotjob, storage\_service, ['\*.png'], public=True)*

*pngtocloud.set\_upstream(plots)* – this is equivalent to *plotflow.add\_edge(plots, pngtocloud)*

Prefect will automatically create edges when it sees dependencies between tasks. For example, there is no explicitly defined edge in the following:

*postmach = ctasks.cluster\_init(postconf)*

*pmStarted = ctasks.cluster\_start(postmach)*

But, there is an existing variable dependency between them (*postmach)*. pmStarted uses postmach as an argument.

Watchout! In a prefect Flow, each line/task has a type or subtype of Prefect Task. Return values from functions are automatically wrapped and unwrapped by Prefect. For instance, cluster and postmach are both being used as “Cluster” types/instances, but in the Flow they are really Prefect Task types. Use of Python typehints can be helpful to the Prefect framework.

##### Prefect Tasks

A task is just a python function with a @task Prefect attribute.

Example: ./workflow/workflows/tasks.py

@task

def save\_to\_cloud(job: Job, service: StorageService, filespecs: list, public=False):

… do some stuff

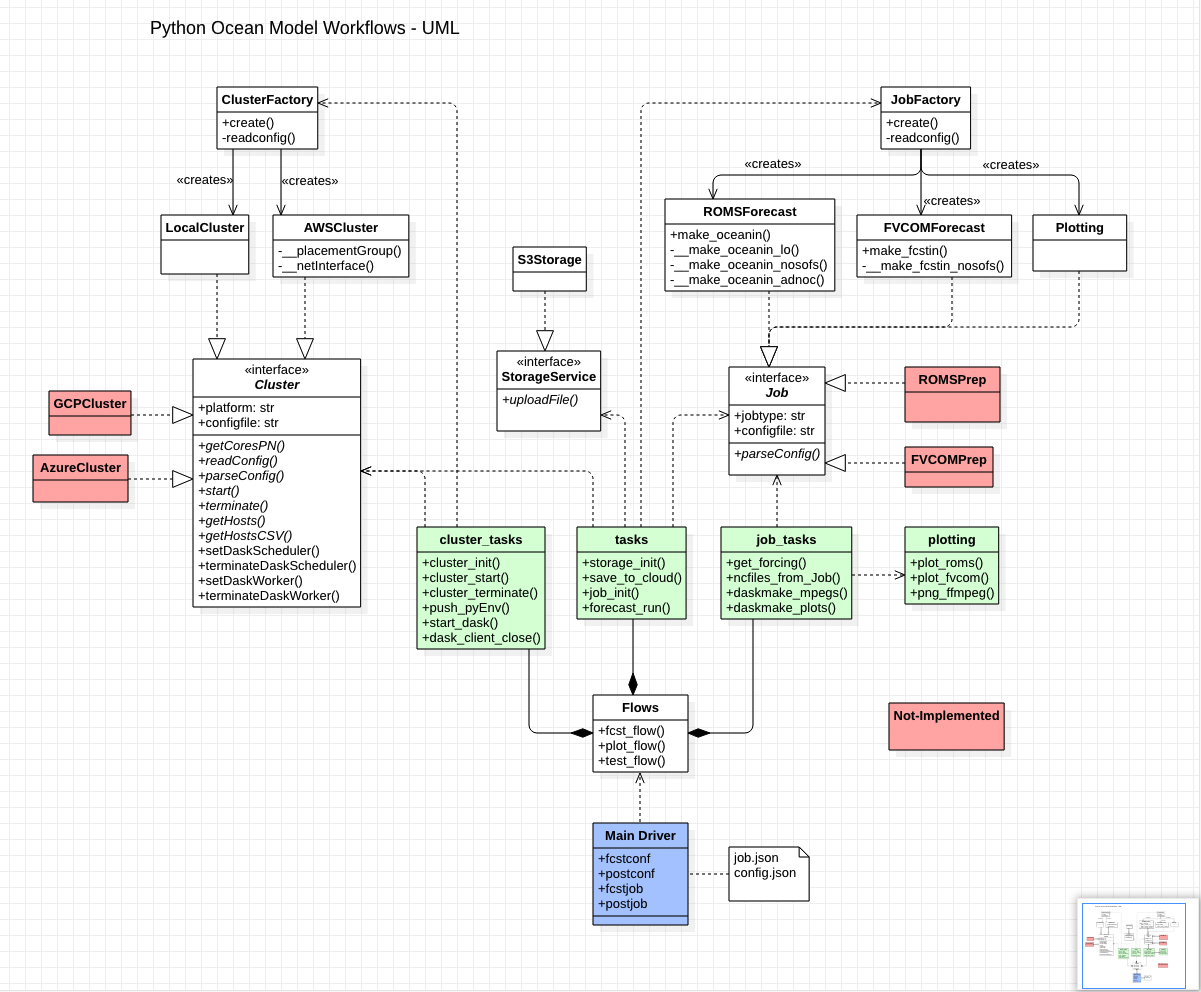
….

service.uploadFile(filename, BUCKET, key, public)

return

A link to the Prefect API documentation is provided in Appendix C

#### Appendix A – Python Workflows UML Class Diagram



#### Appendix B – Models Tested

##### ROMS Models Tested

Model Description Dims DT Decomp (Ops) Filesize Forecast Length

CBOFS Chesapeake Bay (330x289 dt 5) (NtileIxJ 10x14) 59MB/file. NHOURS=48

*NTIMES=34560*

DBOFS Delaware Bay (117x730 dt 5) (NtileIxJ 5x28) 31MB/file NHOURS=48

*NTIMES=34560*

TBOFS Tampa Bay (174x288 dt 5) (NtileIxJ 7x14) 19MB/file NHOURS=48

*6.5 minutes / 24 hour foreast on 4 c5n.18xlarge NTIMES=34560*

*24 minutes / 24 hour forecast on 2 c5n.18xlarge*

GoMOFS Gulf of Maine (1171x775 dt 45) (NtileIxJ 40x28) 703MB/file NHOURS=72

*40 minutes / 24 hour forecast on 4 c5n.18xlarge NTIMES=5760*

CIOFS Cook Inlet Alaska (722x1042 dt 4) (NtileIxJ 26x28) 560MB/file NHOURS=48

*45 minutes / 24 hour forecast on 4 c5n.18xlarge NTIMES=43200*

LiveOcean Northwest US (661x1300 dt 40) (NtileIxJ 14x14) 1848MB/file NHOURS=73

ADNOC Dubai - RPS Proprietary (687x470)

##### Untested ROMS Models

WCOFS West Coast – Experimental/In-Development – missing fixed fields and input files

##### FVCOM Models Tested

Model Description Filesize Forecast Length

NGOFS Northern Gulf of Mexico (174474 CELLS, 41 LVLS) 145MB /file NHOURS=54

*24 minutes/24 fhours on 4 c5n.18xlarge*

NWGOFS Northwest Gulf of Mexico (161518 CELLS, 21 LVLS) 103MB/file NHOURS=48

*16 minutes/24 fhours on 2 c5n.18xlarge*

NEGOFS Northeast Gulf of Mexico (130814 CELLS, 21 LVLS) 83MB/file NHOURS=48

*24 minutes/24 fhours on 2 c5n.18xlarge*

LEOFS Lake Erie (11509 CELLS, 21 LVLS) 12MB/file NHOURS=120

*2.5 minutes/24 fhours on 2 c5n.18xlarge*

SFBOFS San Francisco Bay (102264 CELLS, 21 LVLS) 53MB/file NHOURS=48

*20 minutes/24 fhours on 2 c5n.18xlarge*

LMHOFS Lakes Michigan and Huron (171377 CELLS, 21 LVLS) 173MB/file NHOURS=120

*20 minutes/24 fhours on 2 c5n.18xlarge*

EC2 Pricing on c5n.18xlarge instances as of February 27, 2020: $3.888 per Hour

<https://aws.amazon.com/ec2/pricing/on-demand/>

Example: NGOFS: 54/60 hours \* 4 nodes \* 3.888 = $13.97 per 54 hour forecast.

EC2 Pricing on t3.small instance: $0.0188 per Hour

Example: Always on main node = $3.158 per Week

There may be cheaper options depending on the usage scenario, e.g. Spot instances, dedicated servers.

EFS data storage pricing: $0.30 per GB month.

<https://aws.amazon.com/efs/pricing/>

EC2 EBS data storage pricing: $0.10 per GB month.

<https://aws.amazon.com/ebs/pricing/>

Example: 8 GB gp2 ssd always running = $0.80 per month.

S3 pricing: Storage $0.023 per GB month.

Data transfer out varies, see below link. (Possibly able to use AWS GovCloud (US-East)?)

<https://aws.amazon.com/s3/pricing/>

#### Appendix C – Useful Links

EFA Getting Started:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/efa-start.html>

Prefect:

<https://docs.prefect.io/core/>

<https://docs.prefect.io/api/latest/>

Dask.distributed Client:

<https://distributed.dask.org/en/latest/client.html>

UML Class Diagram Tutorial:

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

#### Appendix D – Notes

python -m SimpleHTTPServer 8000