

IOOS Cloud Sandbox

Coastal Coupling Community of Practice November 2021

Patrick Tripp, Principal Software Engineer

November 2, 2021



Who is the IOOS Cloud Sandbox for?

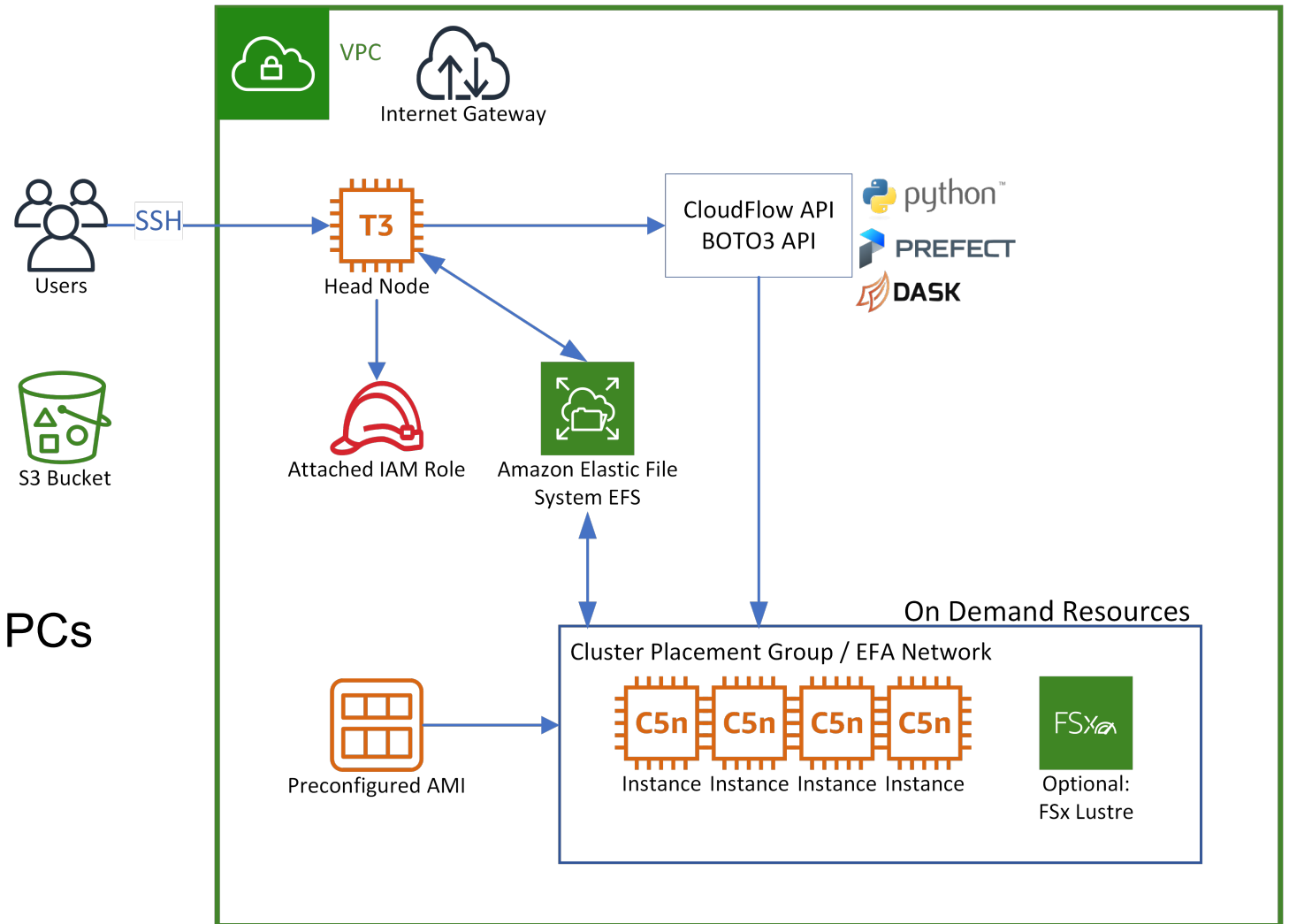
- Is developed with IOOS and others to run regional coastal models
- Multiple models have been setup and tested:
 - NOSOFS ROMS and FVCOM Models
 - LiveOcean – University of Washington
 - WRF/ROMS - ESMF Coupled - Hurricane Irene test case
 - ADCIRC - Hurricane Florence test case
- IOOS Regional centers
- Scientists and researchers
- Universities and students



What is the IOOS Cloud Sandbox?

- Cloud HPC Infrastructure (AWS)
- Software libraries and tools
 - Intel and GCC Compilers
 - NetCDF, MPI, wgrib, etc.
 - Run scripts and workflows
- Environment similar to operational HPCs

- Model analysis and validation
- Research R2O2R
- Operational fail-over
- Hindcasts



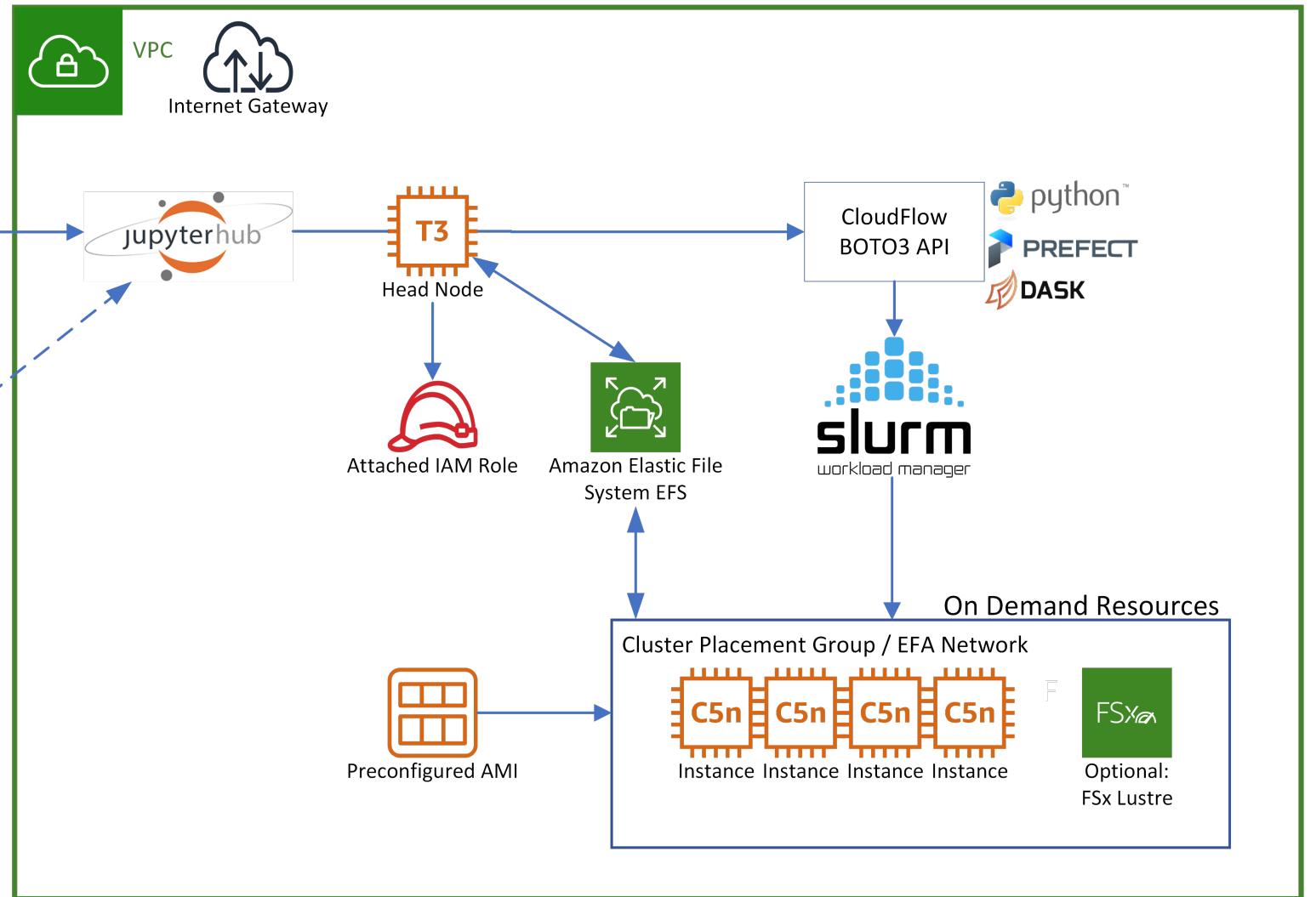
IOOS Cloud Sandbox
Old Version

IOOS Cloud Sandbox Updates

New Version

- JupyterHub head node
- Authentication with Google
- SLURM Workload Manager
- Infrastructure as Code

-  **Terraform**



IOOS Cloud Sandbox
New Version

How to get started!

- Use your own AWS account – self governance
- or have someone else manage the AWS account and resources

- github.com/ioos/Cloud-Sandbox

- 15 minutes to run the setup
 - Creates the AWS infrastructure
 - Installs all of the required software and libraries
 - NetCDF, HDF5, Intel MPI, Intel Compilers, GCC Fortran, Python3, etc.
 - Creates the AMI (Amazon Machine Image) that is reused for the on-demand compute nodes

- github.com/ioos/nosofs-NCO

Currently Supported and Tested Models

- NOSOFS ROMS and FVCOM Models
 - CBOFS, CIOFS, DBOFS, GOMOFS, TBOFS, LEOFS, LMHOFS, NEGOFs, NGOFS, NWGOFS, and SFBOFS
 - Quasi-operational
- LiveOcean
 - Quasi-operational with operational fail-over triggering
- Coupled WRF/ROMS w/ ESMF v8
 - Hurricane Irene test case
- ADCIRC
 - Full test suite in ADCIRC repository
 - Hurricane Florence test case

Currently in Development and Testing

- JupyterHub integration on the head node
 - sign in with Google account
 - or sign in with GitHub account
 - or sign in with other OAuth providers
- SLURM Workload Manager
 - Manage the compute resources
 - Fine grained accounting of usage
- Spack Package Manager



Spack

Jupyter Lab Features

- File explorer
- Terminal
- Text editor
- Interactive notebooks

The screenshot displays the JupyterLab interface. On the left is a file explorer showing a directory structure with files like `cloudflow-test.ipynb`, `hifs-demo.ipynb`, `hifs.config`, `hifs.ipynb`, and `hifs.py`. The central terminal window shows the output of a `conda list` command, listing various packages and their versions. On the right, a notebook cell titled "Load some data" contains Python code that reads a configuration file and loads data from a dataset. The output of the code is displayed below the cell, showing dimensions and coordinates for the loaded data.

```
(base) jupyter-kenny@ip-10-0-1-161:~$ conda list
# packages in environment at /opt/tljh/user:
#
# Name                    Version           Build    Channel
#-----
_libgcc_mutex             0.1              conda_fo conda-forge
_openmp_mutex             4.5              1_llvm  conda-forge
alembic                   1.4.2            pypi_0  pypi
asn1crypto                1.3.0            pypi_0  pypi
async-generator           1.10             pypi_0  pypi
attrs                     19.3.0           pypi_0  pypi
backcall                  0.1.0            pypi_0  pypi
bcrypt                    3.1.7            py37h7b6447c_0
bleach                    3.1.4            pypi_0  pypi
boto3                     1.12.34          pypi_0  pypi
botocore                  1.15.32          pypi_0  pypi
bzip2                     1.0.8            h7b6447c_0
ca-certificates           2020.4.5.1       hecc5488_0 conda-forge
cartopy                   0.17.0           py37h6078e7d_1013 conda-forge
certifi                   2020.4.5.1       pypi_0  pypi
certipy                   0.1.3            pypi_0  pypi
cffi                      1.14.0           py37h2e261b9_0
cftime                    1.1.1.1          py37heb32a55_0
chardet                   3.0.4            py37_1003
click                     7.1.1            pypi_0  pypi
cloudflow                 1.0.1            pypi_0  pypi
file                       1.3.0            pypi_0  pypi
package-handling          2.0              pypi_0  pypi
phy                       4.8.3            py37hc8dfbb8_1 conda-forge
sphinx                    1.6.0            py37h7b6447c_0
sphinx-panels             0.3.31           pypi_0  pypi
sphinxcontrib             2.8              py37h1ba5d50_0
sphinxcontrib-jupyterlab 7.69.1           hbc83047_0

config
"BOBTYP" : "plotting",
"OFS" : "cbofs",
"CDATE" : "20200421",
"HH" : "00",
"INDIR" : "/com/nos",
"OUTDIR" : "/com/nos/plotting",
"PARS" : ["temp", "zeta", "w", "salt"],
"BUCKET" : "ioos-cloud-sandbox",
"BUCKETFLDR" : "public/nosofs/plots",
"FSPEC" : "nos.cbofs.fields.f*t00z.nc"
```

```
Load some data
[8]: fp = 'hifs.config'
with open(fp, 'rb') as f:
    config = json.load(f)
[9]: if config['CDATE'] == 'today':
    CDATE = datetime.date.today().strftime("%Y%m%d")
else:
    CDATE = config['CDATE']
OFS = config['OFS']
fspec = config['FSPEC']
indir = f"{config['INDIR']}/{OFS}.{CDATE}"
outdir = f"{config['OUTDIR']}/{OFS}.{CDATE}"
ds = xr.open_mfdataset(f'{indir}/{fspec}', decode_times=False, combine='by_coords')
ds
[9]: xarray.Dataset
Dimensions: (boundary: 4, eta_psi: 290, eta_rho: 291, eta_u: 291, eta_v: 290,
ocean_time: 48, s_rho: 20, s_w: 21, tracer: 3, xi_psi: 331, xi_rho: 332,
xi_u: 331, xi_v: 332)
Coordinates:
lon_v          (eta_v, xi_v) float64 dask.array<chunksize=(290, 332), meta=n...
lat_rho        (eta_rho, xi_rho) float64 dask.array<chunksize=(291, 332), meta=n...
lon_rho        (eta_rho, xi_rho) float64 dask.array<chunksize=(291, 332), meta=n...
lat_u          (eta_u, xi_u) float64 dask.array<chunksize=(291, 331), meta=np...
lat_v          (eta_v, xi_v) float64 dask.array<chunksize=(290, 332), meta=n...
lat_psi        (eta_psi, xi_psi) float64 dask.array<chunksize=(290, 331), meta=n...
lon_psi        (eta_psi, xi_psi) float64 dask.array<chunksize=(290, 331), meta=n...
lon_u          (eta_u, xi_u) float64 dask.array<chunksize=(291, 331), meta=np...
s_w            (s_w) float64 -1.0 -0.95 -0.9 ... -0.1 -0.05 0.0
s_rho          (s_rho) float64 -0.975 -0.925 ... -0.075 -0.025
ocean_time     (ocean_time) float64 1.358e+08 1.358e+08 ... 1.36e+08
Data variables:
(97)
Attributes: (34)
```

The screenshot shows the JupyterHub login page. It features the JupyterHub logo at the top left and a "Login" button at the top right. Below the logo is a "Sign In" section with a "Username:" label and a text input field containing the text "patrick". Below the username field is a "Password:" label and a password input field with a toggle eye icon. A "Sign In" button is located below the password field. At the bottom of the sign-in section, there is a link that says "Don't have an user? [Signup!](#)".

Links

GitHub Repositories

<https://github.com/ioos/Cloud-Sandbox>

<https://ioos.github.io/Cloud-Sandbox>

<https://github.com/ioos/nosofs-NCO>

- Terraform, Python, and BASH tools
- API documentation for above
- copy of NOAA operational version with local changes

Questions and Comments