PODPAC: The Easy Way to Analyze Earth Science Data in the Cloud

J. Bieszczad *, M. Ueckermann,* M. Shapiro *, D. Callender *, D. Sullivan *, D. Entekhabi †

*Creare LLC, † Massachusetts Institute of Technology

Motivation

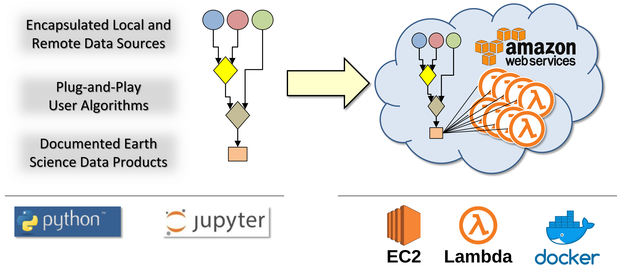
- There are significant benefits to cloud-based analysis and analytics:
  - Low cost and effort to access data stored on cloud
  - On-demand access to massive compute resources
  - Lower maintenance cost
- However, there are also significant barriers to adopting a new cloud-based workflow:
  - Vendor-specific expertise and knowledge
  - Multiple approaches to achieve the same goal
  - Administrative barriers to fund cloud infrastructure
  - Cloud infrastructure maintenance
  - Non trivial to transfer existing workflows to cloud

Our Solution

Pipeline for Observational Data Processing, Analysis, and Collaboration

Readily develop integrated geospatial analyses and analytics on your workstation

Seamlessly transition to scalable, massively distributed processing on the cloud

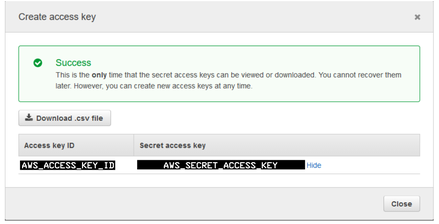


Open Source Development

- PODPAC is free and open-source software available at https://podpac.org/

Easy 1,2,3 to the Cloud

1. Create Amazon Web Services account
2. Using AWS console, create and record:
   AWS_ACCESS_KEY_ID
   AWS_SECRET_ACCESS_KEY



3. Install PODPAC and build resources →

```
from podpac import settings
from podpac.managers import aws

# Specify AWS credentials in settngs and save
podpac.settings['AWS_ACCESS_KEY_ID'] = AWS_ACCESS_KEY_ID
podpac.settings['AWS_SECRET_ACCESS_KEY'] = AWS_SECRET_ACCESS_KEY
podpac.settings.save()

# Specify the name for the S3 Bucket
settings["S3_BUCKET_NAME"] = "podpac-lambda-function-name"  # Must be globally unique (AWS-wide!)

# Create a PODPAC AWS Lambda node
node = aws.Lambda()

# Create all the AWS resources
# This can take up to 15-20 seconds if all resources need to be created
node.build()
```

Cloud resources created and configured automatically

Generic PODPAC Lambda (compute) · S3 (storage) · IAM (permissions) · API Gateway (HTTP API)

Example transitioning local workflow to cloud

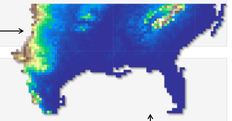
```
# Import DEM datasource Node
from podpac.datalib.terraintiles import TerrainTiles

# Create a processing pipeline
dem = TerrainTiles(zoom=1)
dem2 = podpac.algorithm.Arithmetic(dem=dem, eqn='dem**2')

# Create coordinates
coordinates = podpac.Coordinates([podpac.clinspace(-42, -41, 512, 'lat'),
                                  podpac.clinspace(-72, -73, 512, 'lon')])

# Evaluate locally
out_xarray = dem2.eval(coordinates)

# Evalute in the cloud
dem2cloud = aws.Lambda(
    source=dem2,            # Any PODPAC Node
    download_result=True  # If False, results will reside on S3
)
out_xarray = dem2cloud.eval(coordinates)
```

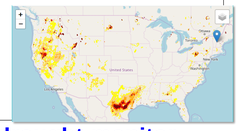


Example WMS feed using Leaflet in browser

```
<!DOCTYPE html>
<html>
<head>
  <!-- Leaflet -->
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.5.1/dist/leaflet.css" integrity="
  sha512-xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIptF7tcCzusKFjFw2yuvEpDL9wQ==
  crossorigin=""/>
  <script src="https://unpkg.com/leaflet@1.5.1/dist/leaflet.js" integrity="
  sha512-GffPMF3RvMeYyc1LWMHtK8EbPv0iNZ8/oTtHPx9/cc2ILxQ+u905qIwdpULaqDkyBKgOaB57QTMg7ztg8Jm2Og==
  crossorigin=""></script>
</head>
<body>
  <!-- Map Page -->
  <div id="map" class="map" style="height:800px;"></div>
  <script>
    // API URL from node._function_api_url in Python
    var SMAPWMS = L.tileLayer.wms(
      "https://ps1dfpoecf.execute-api.us-east-1.amazonaws.com/prod/eval/?",
      {
        transparency: true, opacity: 0.95, format: 'image/png', time: "2020-01-01",
        layers: "https://podpac-drought-monitor-s3.s3.amazonaws.com/pipeline_category.json"
      });
    var map = L.map('map',
    {
      center: [42, -100.0], zoom: 4,
      layers: [SMAPWMS]
    });
  </script>
</body>
</html>
```

PODPAC API Gateway

Any PODPAC Pipeline

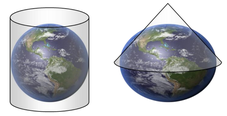


https://creare-com.github.io/podpac-drought-monitor

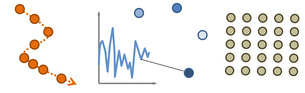
Benefits

- Automate geo-data wrangling for **integrated analyses of disparate data sources** in a plug-and-play manner
- Enable data scientists to **easily transition workstation analyses to massively distributed processing** on Amazon Web Services (AWS)
- Facilitate **generation and sharing** of reproducible and documented earth science data products and algorithms
- Automated data wrangling handles differences in geospatial CRS, projections, resolution, formats, etc.

Geospatial CRS and Projections



Data Structures



GeoTIFF · HDF · netCDF · OPeNDAP · GRIB · CSV · Zarr

- **JSON** metadata enables direct **deployment** and execution of PODPAC algorithm pipelines **on AWS**
- PODPAC-enabled "**serverless**" AWS Lambda functions avoid provisioning and maintenance of cloud servers
- PODPAC Lambda functions automatically **scale up to 1024 parallel computational processes**
- **Processing** on AWS "**close to data storage**" improves performance and avoids costly egress charges

Acknowledgment

- This research is supported by NASA under SBIR Phase II Contract No. 80NSSC18C0061