Microsoft

# Scalable Geospatial Analysis

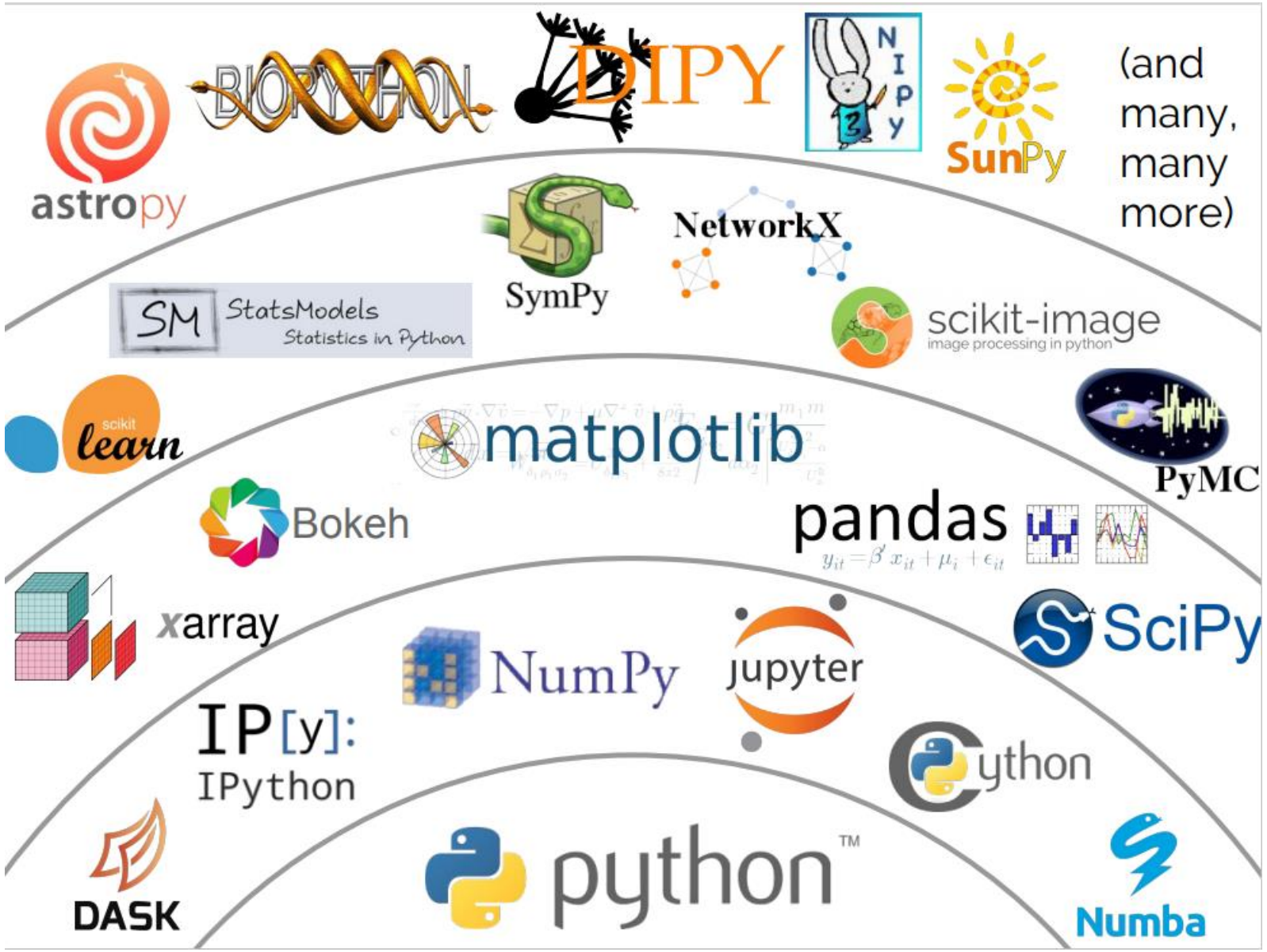
## from a Dask point of view

Tom Augspurger
Microsoft AI For Earth
May 19, 2021

# A Planetary Computer for a Sustainable Future

# Planetary Computer

- **Data Catalog**: Petabytes of data in Blob Storage
- **API**: STAC-based APIs for search and discovery
- **Compute**: JupyterHub / Dask
- **Applications**: Putting the data to use




Jake VanderPlas

*The Unexpected Effectiveness of Python in Science*

PyCon 2017

# STAC – the Spatial Temporal Asset Catalog

# STAC

I need…

this data

*covering*

this region

*over*

this time period

# STAC

I need...

# Landsat 8 Collection 2 Level 2

*covering*

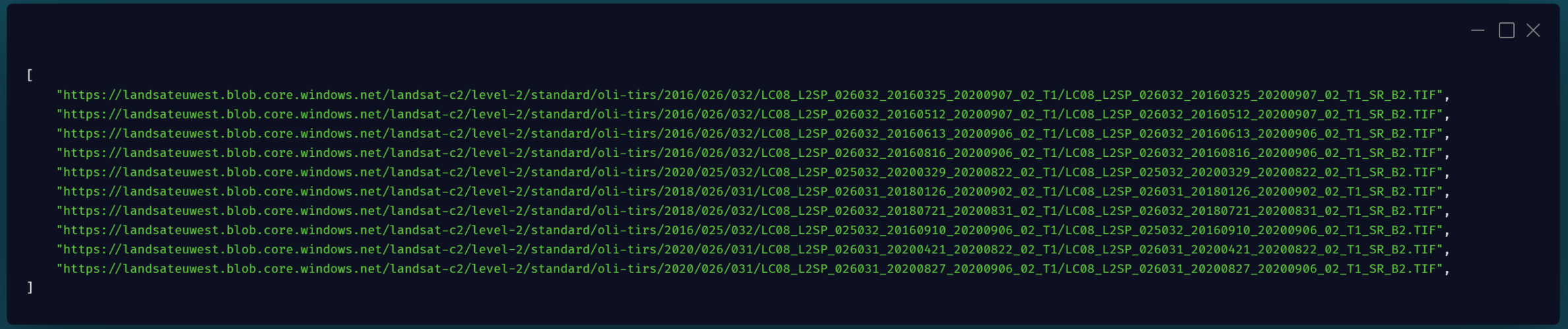
(-93.0, 40.60, -91.7, 41.6)

*over*

2016-01-01/2020-12-31

# STAC

The cloud providers host data, but…



```
[
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2016/026/032/LC08_L2SP_026032_20160325_20200907_02_T1/LC08_L2SP_026032_20160325_20200907_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2016/026/032/LC08_L2SP_026032_20160512_20200907_02_T1/LC08_L2SP_026032_20160512_20200907_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2016/026/032/LC08_L2SP_026032_20160613_20200906_02_T1/LC08_L2SP_026032_20160613_20200906_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2016/026/032/LC08_L2SP_026032_20160816_20200906_02_T1/LC08_L2SP_026032_20160816_20200906_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2020/025/032/LC08_L2SP_025032_20200329_20200822_02_T1/LC08_L2SP_025032_20200329_20200822_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2018/026/031/LC08_L2SP_026031_20180126_20200902_02_T1/LC08_L2SP_026031_20180126_20200902_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2018/026/032/LC08_L2SP_026032_20180721_20200831_02_T1/LC08_L2SP_026032_20180721_20200831_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2016/025/032/LC08_L2SP_025032_20160910_20200906_02_T1/LC08_L2SP_025032_20160910_20200906_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2020/026/031/LC08_L2SP_026031_20200421_20200822_02_T1/LC08_L2SP_026031_20200421_20200822_02_T1_SR_B2.TIF",
    "https://landsateuwest.blob.core.windows.net/landsat-c2/level-2/standard/oli-tirs/2020/026/031/LC08_L2SP_026031_20200827_20200906_02_T1/LC08_L2SP_026031_20200827_20200906_02_T1_SR_B2.TIF",
]
```

A bunch of URLs to blob storage isn’t the easiest to work with

# STAC

```
>>> stac = pystac_client.Client.open(
...     "https://planetarycomputer.microsoft.com/api/stac/v1"
... )

>>> items = stac.search(
...     collections=["landsat-8-c2-l2"],      # this data
...     bbox=(-92.36, 40.66, -92.20, 40.80)   # covering this area
...     datetime="2016-01-01/2020-12-31",     # over this time
... ).items_as_collection()
```

# STAC

Now we have objects with URLs to blob storage! But...

```
>>> %time data = rasterio.open(items[0].assets['SR_B2'].href)
CPU times: user 26.9 ms, sys: 4.68 ms, total: 31.6 ms
Wall time: 607 ms
```

# STAC

- Reading the metadata of a single item takes ~1s
- Our search returned about 350 items
- 5 minutes just to open a dataset, without actually downloading data?

# <Pop Quiz!>

# Pop Quiz!

*True or False…* Dask is Lazy

# Pop Quiz!

*True or False…* Dask is Lazy

True*

(or False*)

# Pop Quiz!

*True or False…* Dask is Lazy

```
>>> da.from_zarr( ... )      # reads .zmetadata
>>> dd.read_parquet( ... )   # reads _common_metadata, _metadata
>>> dd.read_csv( ... )       # reads 10,000 bytes, infers
```

# Pop Quiz!

*True or False…* **Dask is Lazy**

- Dask collections do *just enough* actual I/O to construct the **metadata**
- Subsequent operations are lazy

# </Pop Quiz>

# STAC

Back to STAC, what's in this Item?

- `proj:epsg 32615`
- `proj:shape [7891, 7771]`
- `proj:transform [29.99, 0.0, 346800.0, 0.0, -29.99, 4582500.0]`
- `eo:bands [{'name': 'SR_B3', 'common_name': 'green', 'gsd': 30, 'center_wavelength': 0.56, 'full_width_half_max': 0.06}`
- `datetime 2016-01-01T00:00Z`

In other words… *just enough* to construct the **metadata**

# stackstac

Create a DataArray from STAC items *without opening any files.*

```
>>> stackstac.stack(items)
<xarray.DataArray 'stackstac-179c4' (time: 174, band: 19, y: 13173, x: 13463)>
dask.array<..., shape=(174, 19, 13173, 13463), chunksize=(1, 1, 1024, 1024)>
Coordinates: (12/27)
  * time                     (time) datetime64[ns] 2016-11-15T16:53:42.80 ...
    id                       (time) <U40 'LC08_L2SP_026031_20160105_20200 ...
  * band                     (band) <U9 'QA_PIXEL' 'QA_RADSAT' ... 'ST_QA'
  * x                        (x) float64 3.453e+05 3.453e+05 ... 7.491e+05
  * y                        (y) float64 4.741e+06 4.741e+06 ... 4.346e+06
    platform                 <U9 'landsat-8'
    ...                       ...
    title                    (band) <U46 'Pixel Quality Assessment Band' ...
    gsd                      (band) object None None None ... 30.0 30.0 30.0
    common_name              (band) object None None 'coastal' ... None None
    center_wavelength        (band) object None None 0.44 ... None None None
    full_width_half_max      (band) object None None 0.02 ... None None None
    epsg                     int64 32615
Attributes:
    spec:           RasterSpec(epsg=32615, bounds=(345284.98240125144, 434578...
    crs:            epsg:32615
    transform:      | 30.00, 0.00, 345284.98|\n| 0.00,-30.00, 4740921.41|\n| ...
    resolution_xy:  (29.99608916699257, 29.99614940315749)
```

# Demo

# Summary

- Many groups facing similar challenges, especially around scaling
- We benefit from domain-specific libraries using general-purpose, shared data structures, and standards like STAC
- Geospatial is cool?

Microsoft

https://planetarycomputer.microsoft.com

https://github.com/TomAugspurger/scalable-geospatial