

The Planetary Computer: Putting Global-Scale Geospatial Data to Work for Conservation and Sustainability

Dan Morris

Microsoft AI for Earth

dan@microsoft.com

aka.ms/aiforearth

aka.ms/dmorris

gratuitous
pictures of
animals that
aren't related
to my talk



AI for Earth is three things...



Grants
aka.ms/ai4egrants



ML stuff
aka.ms/ai4etech



Planetary Computer
aka.ms/planetarycomputer

AI for Earth: our grants program

850+
grants

120+
countries

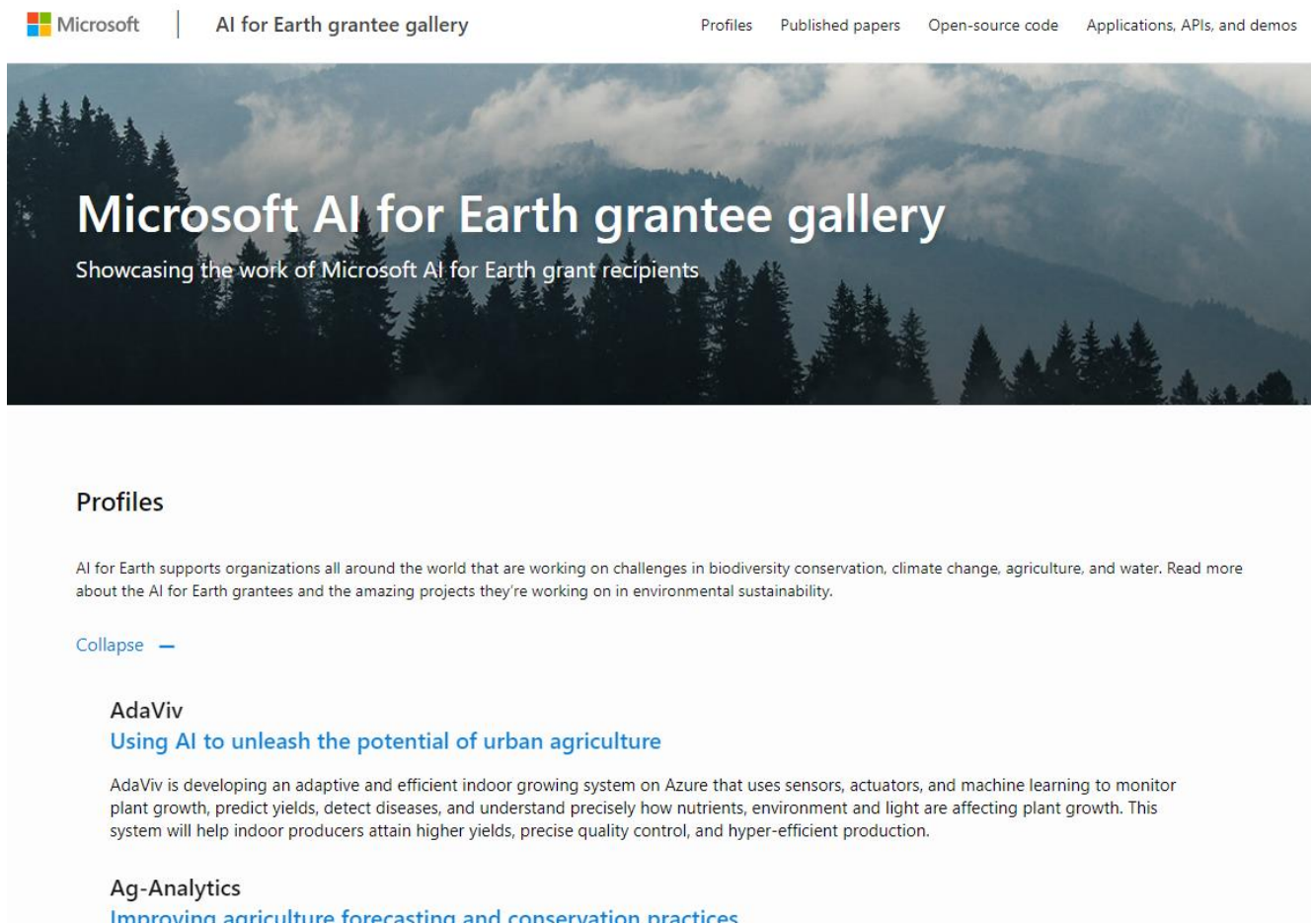
aka.ms/ai4egrants

GEO-Microsoft RFP



aka.ms/geogrant

AI for Earth: grantee stories



aka.ms/ai4egrantees

ML stuff AI for Earth builds:

Using ML to help conservation scientists spend *less time clicking stuff*,
and *more time doing conservation*.

Accelerating **camera trap** surveys

github.com/microsoft/cameratraps



Accelerating **aerial wildlife** surveys

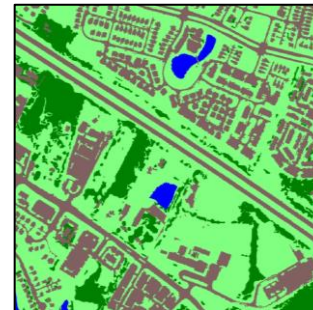
github.com/microsoft/aerial_wildlife_detection

github.com/microsoft/arcticseals



Accelerating **land cover** surveys

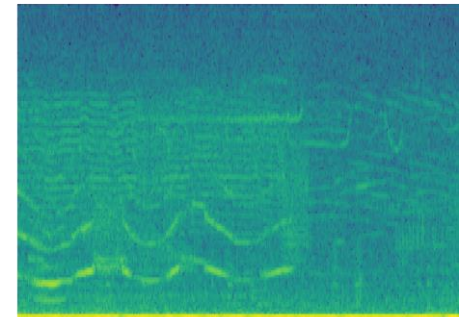
aka.ms/landcovermapping



Accelerating **acoustic wildlife** surveys

github.com/microsoft/belugasounds

github.com/microsoft/multi_species_bioacoustic_classification



All the fun animal-related work in one place

aka.ms/biodiversitysurveys

Why a Planetary Computer?



Environmental sustainability depends on huge geospatial data sets, especially satellite imagery and climate data.



Working with geospatial data is a pain unless you have a PhD in remote sensing.



Working with very large data is a pain unless you have a PhD in distributed computing.



The people at the front lines of conservation usually have neither of the above.

The Planetary Computer *platform* puts key environmental datasets alongside processing tools and a managed compute environment to lower the access barrier for sustainability practitioners.

Planetary Computer *applications* put that platform to work for environmental decision-making.

Planetary Computer Components



Geospatial **data** (currently ~25PB)



Data querying and processing **APIs**



Compute environment (aka "Hub") for analytic workflows



(Partner) **applications** that put all of the above to work for sustainability

Planetary Computer Data



Remote sensing data

- Landsat 4, 5, 7, 8
- Sentinel-1, -2, -3, -5P
- GOES-16, -17
- MODIS, NAIP, ASTER, HLS
- NASADEM, ALOS DEM, Copernicus DEM



Weather/climate data

- GFS, HRRR, ISD, GHE, TerraClimate, Daymet, CMIP6



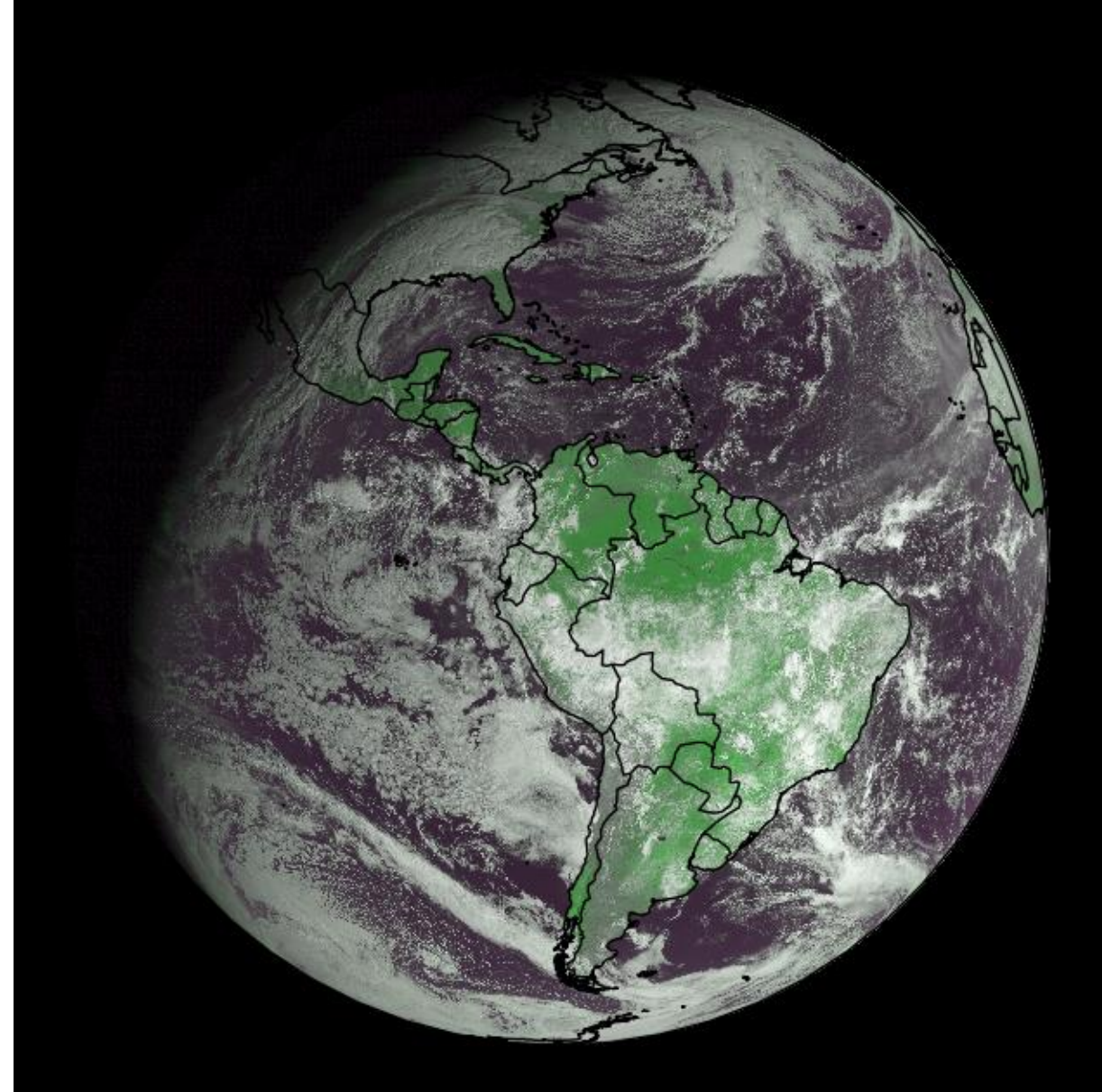
Land cover data

- USGS GAP, Esri 10m, JRC Water, NLCD



Biodiversity data

- GBIF, NatureServe MoBI, USFS FIA



planetarycomputer.microsoft.com/catalog

That gets us to a super-slick pile of files...

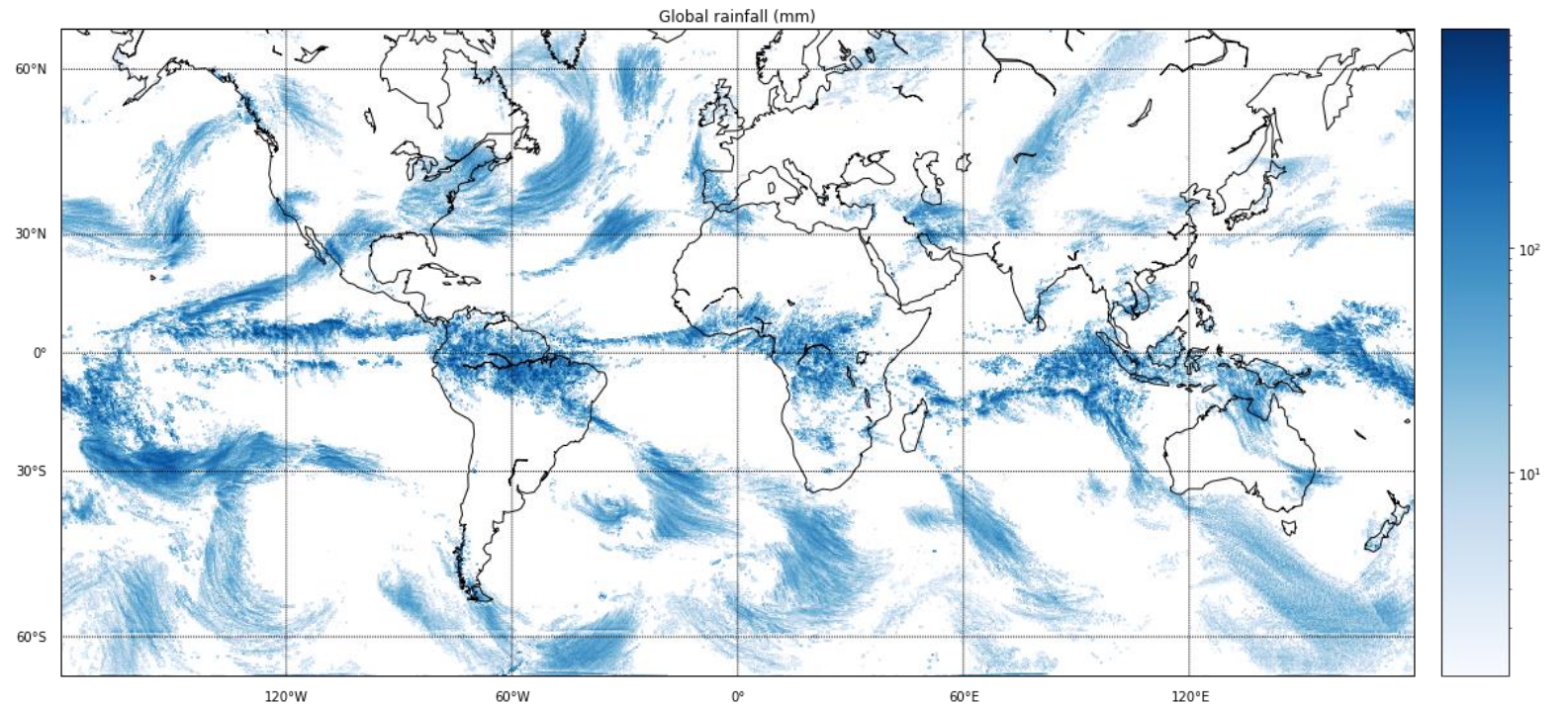
Planetary Computer APIs



Spatiotemporal queries
("find me all the images that overlap with Wyoming in 2012")



Image processing APIs: e.g. resampling to new grids, blending images



That gets us to a super-slick pile of
files that's easy to query...

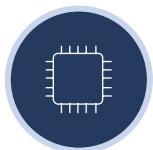
Planetary Computer Hub



JupyterLab front-end



Access to Planetary Computer Data and APIs,
and to user data on Azure storage



Distributed processing (without having to
know what “distributed processing” is)

planetarycomputer.microsoft.com/compute

```
sentinel-2-l2a-example.ipynb x
Python 3

from URLs via HTTP GET operations.

For example, here we use rasterio to render the image data over our area of
interest:

Render our AOI from this image

[8]: import rasterio
      from rasterio import windows
      from rasterio import features
      from rasterio import warp

      import numpy as np
      from PIL import Image

[9]: with rasterio.open(signed_href) as ds:
      aoi_bounds = features.bounds(area_of_interest)
      warped_aoi_bounds = warp.transform_bounds('epsg:4326', ds.crs, *aoi_bounds)
      aoi_window = windows.from_bounds(transform=ds.transform, *warped_aoi_bounds)
      band_data = ds.read(window=aoi_window)

rasterio gives us data band-interleave format; transpose to pixel interleave,
and downscale the image data for plotting.

[10]: img = Image.fromarray(np.transpose(band_data, axes=[1, 2, 0]))
      w = img.size[0]; h = img.size[1]; aspect = w/h
      target_w = 800; target_h = (int)(target_w/aspect)
      img.resize((target_w, target_h), Image.BILINEAR)

[10]: 
```

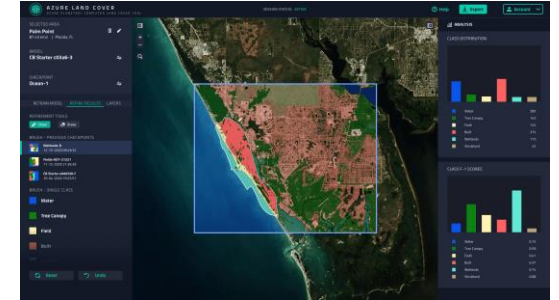
That gives us a super-slick pile of files that's easy to query, and the power to compute over it...

Some key launch applications...



Land use and land cover assessment

Accelerating geospatial image analysis with cloud-based AI, in partnership with Development Seed



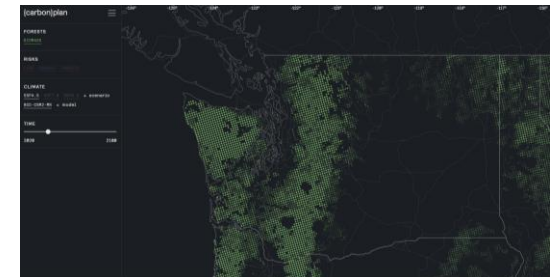
Deforestation risk analysis

Estimating at-risk areas in the Amazon, in partnership with Imazon and Fundo Vale



Prioritizing carbon offset projects

Leveraging climate and fire risk models to inform decisions about forest protection projects, in partnership with CarbonPlan



planetarycomputer.microsoft.com/applications

The Planetary Computer: Putting Global-Scale Geospatial Data to Work for Conservation and Sustainability

Dan Morris
Microsoft AI for Earth
dan@microsoft.com
aka.ms/aiforearth
aka.ms/dmorris

