Google Cloud

NOAA Public Datasets on Google Cloud

01/28/2024

Public Datasets on Google Cloud 01

Accessing and Using Public Datasets 02

Use Cases and Journeys 03

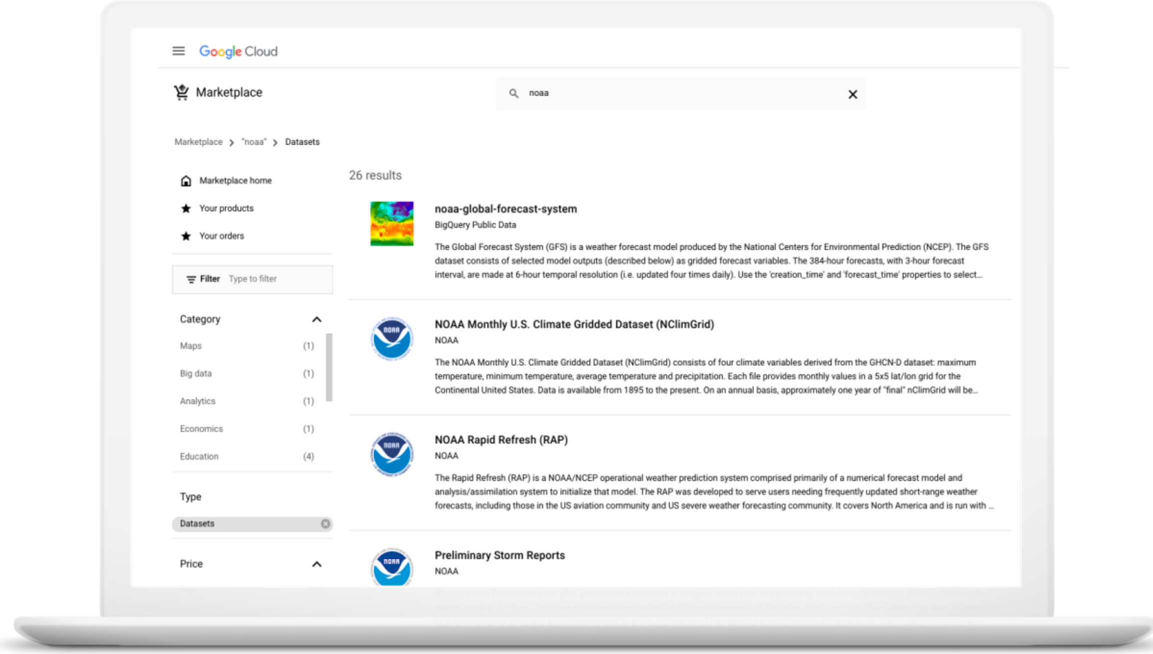
01

Public Datasets on Google Cloud

Google Cloud Public Datasets is a carefully curated and (mostly) Google managed dataset catalog from various sources all around the world, including weather data, shopping data, crypto, and even Google’s own Search data.

Datasets in Marketplace

- Google Cloud Marketplace is the source of truth for datasets in GCP
- Can search and filter through what’s available
- No login required to browse dataset entries
- All consumption of raw (non-tabular, bucket) data is free
- BigQuery tabular data is charged per query



Category	
Maps	(1)
Big data	(1)
Analytics	(1)
Economics	(1)
Education	(4)

26 results

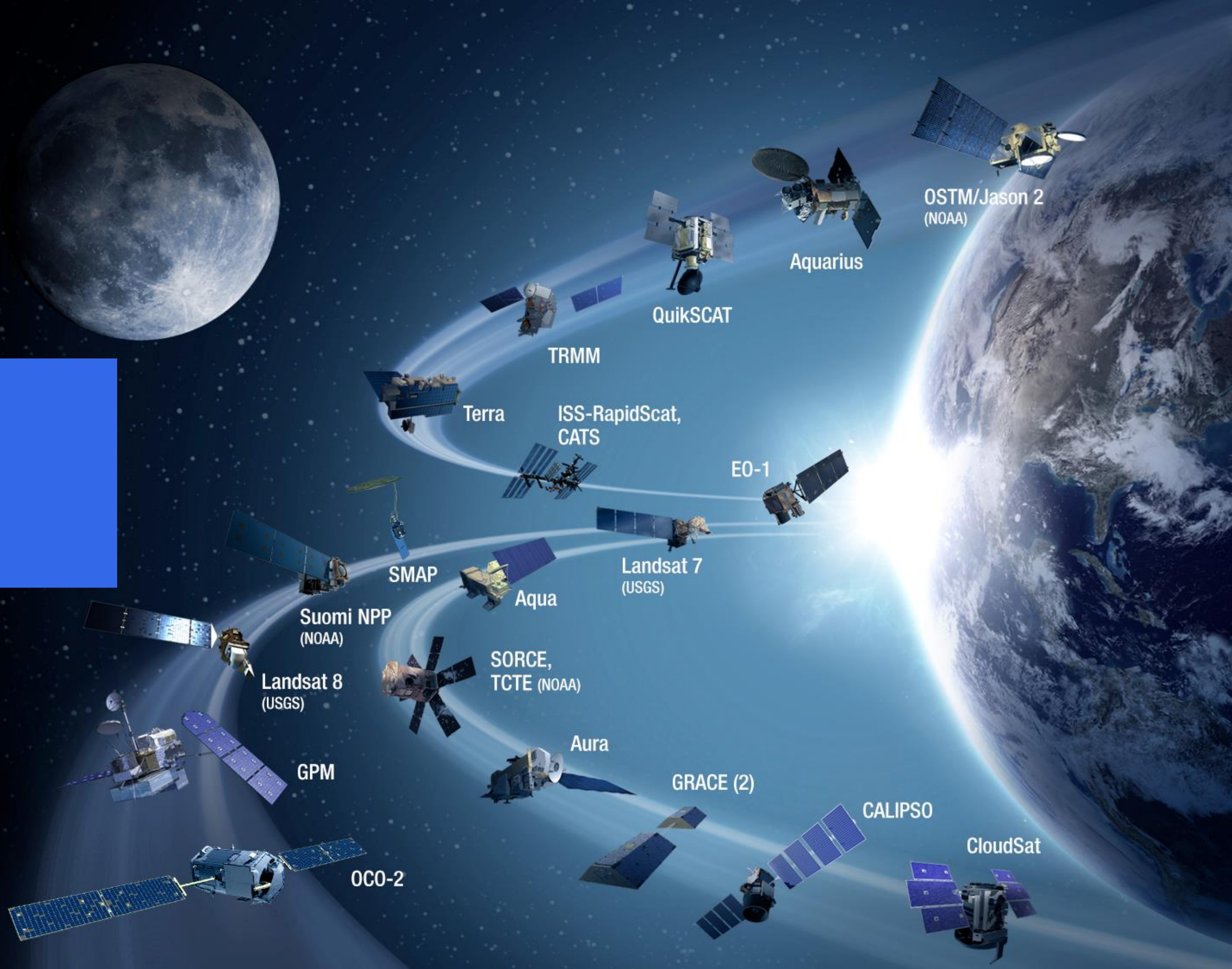
noaa-global-forecast-system
BigQuery Public Data
The Global Forecast System (GFS) is a weather forecast model produced by the National Centers for Environmental Prediction (NCEP). The GFS dataset consists of selected model outputs (described below) as gridded forecast variables. The 384-hour forecasts, with 3-hour forecast interval, are made at 6-hour temporal resolution (i.e. updated four times daily). Use the 'creation_time' and 'forecast_time' properties to select...

NOAA Monthly U.S. Climate Gridded Dataset (NClimGrid)
NOAA
The NOAA Monthly U.S. Climate Gridded Dataset (NClimGrid) consists of four climate variables derived from the GHCN-D dataset: maximum temperature, minimum temperature, average temperature and precipitation. Each file provides monthly values in a 5x5 lat/lon grid for the Continental United States. Data is available from 1895 to the present. On an annual basis, approximately one year of "final" nClimGrid will be...

NOAA Rapid Refresh (RAP)
NOAA
The Rapid Refresh (RAP) is a NOAA/NCEP operational weather prediction system comprised primarily of a numerical forecast model and analysis/assimilation system to initialize that model. The RAP was developed to serve users needing frequently updated short-range weather forecasts, including those in the US aviation community and US severe weather forecasting community. It covers North America and is run with ...

Preliminary Storm Reports
NOAA

An explosion of satellite data



Source: NASA

70+ Petabytes

Growing daily

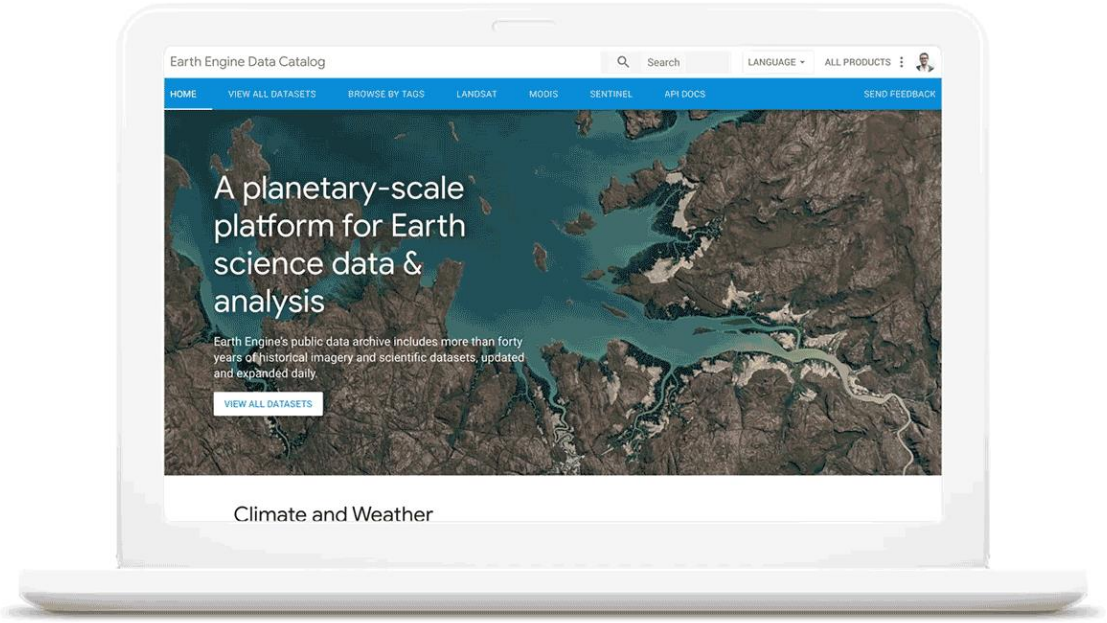
1 Petabyte

Monthly growth rate

700+

Curated datasets

Continuously updated in near real-time



Earth Engine Data Catalog

A planetary-scale platform for Earth science data & analysis

Earth Engine's public data archive includes more than forty years of historical imagery and scientific datasets, updated and expanded daily.

Climate and Weather

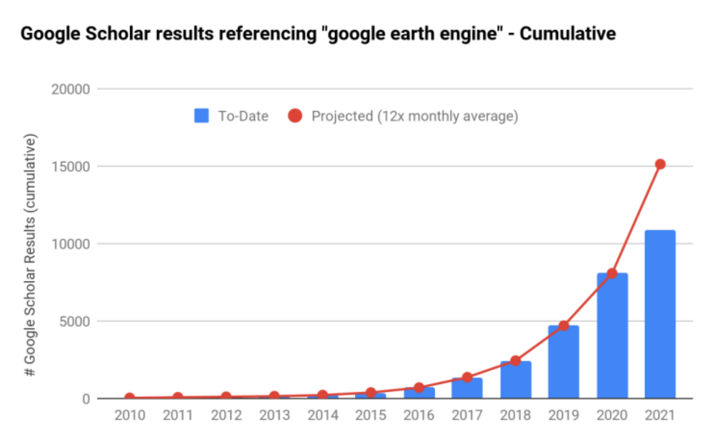
developers.google.com/earth-engine/datasets/

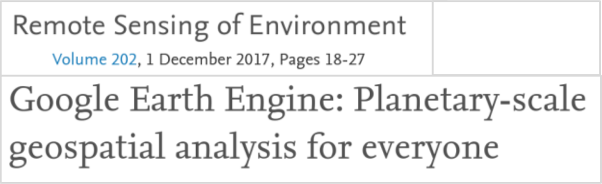
400k+

Scientists around the world

8,000+

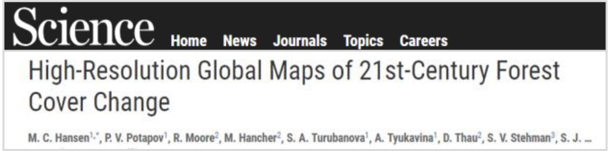
Scientific papers


Google Scholar results referencing "google earth engine" - Cumulative


Remote Sensing of Environment

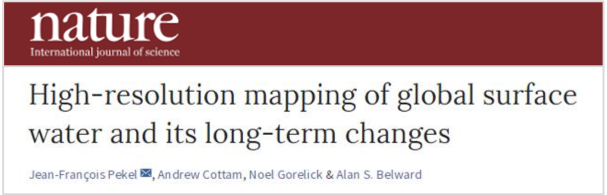
Volume 202, 1 December 2017, Pages 18-27

Google Earth Engine: Planetary-scale geospatial analysis for everyone



High-Resolution Global Maps of 21st-Century Forest Cover Change

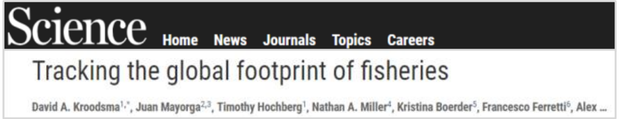
M. C. Hansen[1,*], P. V. Potapov[1], R. Moore[2], M. Hancher[2], S. A. Turubanova[1], A. Tyukavina[1], D. Thau[2], S. V. Stehman[3], S. J. ...


nature

International journal of science

High-resolution mapping of global surface water and its long-term changes

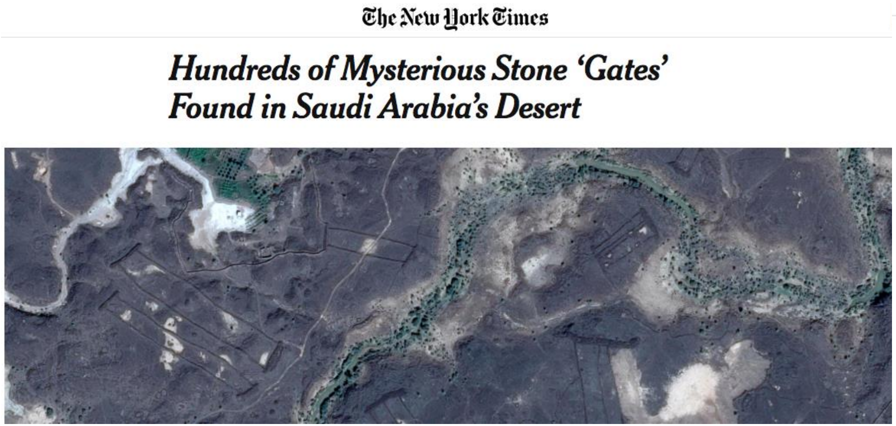
Jean-François Pekel ✉, Andrew Cottam, Noel Gorelick & Alan S. Belward



Tracking the global footprint of fisheries

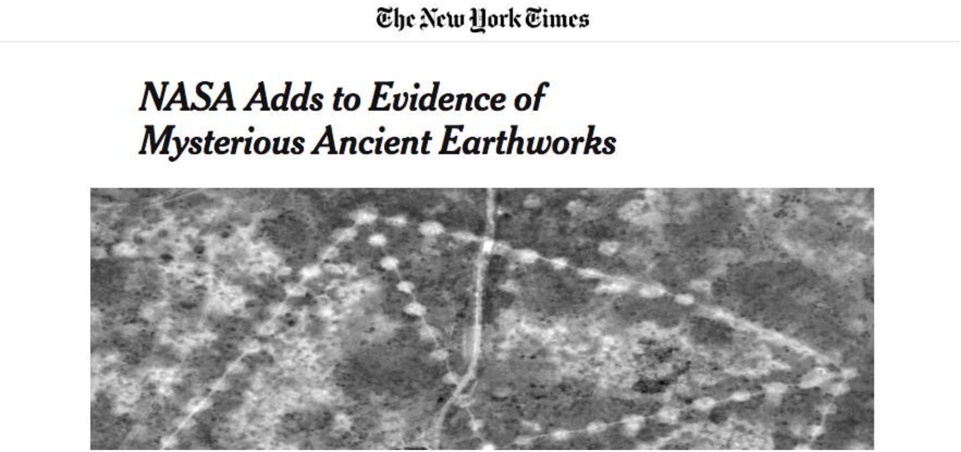
David A. Kroodsma[1,*], Juan Mayorga[2,3], Timothy Hochberg[1], Nathan A. Miller[4], Kristina Boerder[5], Francesco Ferretti[6], Alex ...

Google Earth has helped users find amazing things



The New York Times

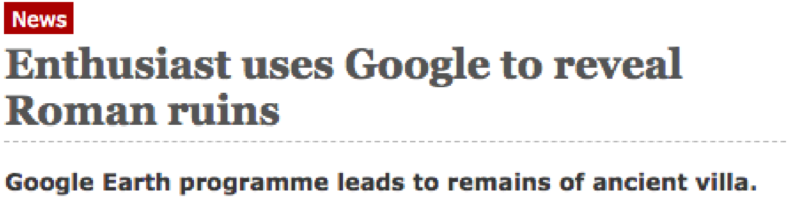
Hundreds of Mysterious Stone 'Gates' Found in Saudi Arabia's Desert



The New York Times

NASA Adds to Evidence of Mysterious Ancient Earthworks

How Google Earth helped find Mozambique's lost forest of Mount Mabu | video

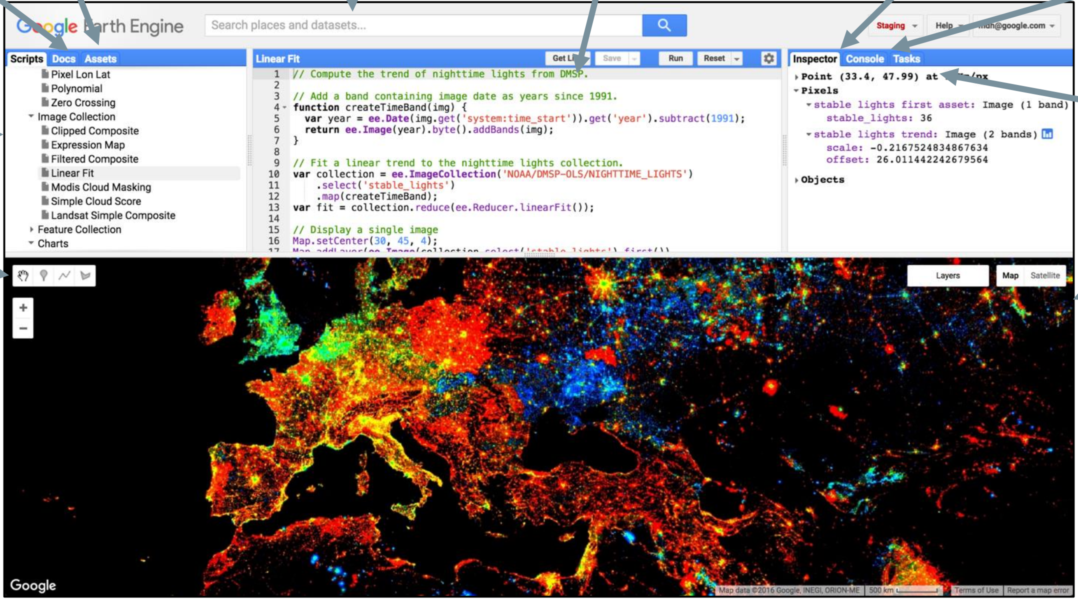


News

Enthusiast uses Google to reveal Roman ruins

Google Earth programme leads to remains of ancient villa.

What is Earth Engine? | Code Editor



```
// Compute the trend of nighttime lights from DMSP.

// Add a band containing image date as years since 1991.
function createTimeBand(img) {
  var year = ee.Date(img.get('system:time_start')).get('year').subtract(1991);
  return ee.Image(year).byte().addBands(img);
}

// Fit a linear trend to the nighttime lights collection.
var collection = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS')
    .select('stable_lights')
    .map(createTimeBand);
var fit = collection.reduce(ee.Reducer.linearFit());

// Display a single image
Map.setCenter(30, 45, 4);
Map.addLayer(ee.Image(collection.select('stable_lights').first())
```

```
Point (33.4, 47.99) at
Pixels
  stable lights first asset: Image (1 band)
    stable_lights: 36
  stable lights trend: Image (2 bands)
    scale: -0.2167524834867634
    offset: 26.011442242679564
Objects
```

API Docs

Your Assets

Search

Your Code

Data Inspector

Output Console

Your scripts, Example scripts

Batch Tasks

Drawing Tools

Output Map

code.earthengine.google.com

Featured NOAA datasets

Global Summary of the Day (GSOD)

A dozen daily averages computed from global hourly station data, covering 1929 to present.

Next Generation Radar (NEXRAD)

High-resolution S-band Doppler weather radars operated by the National Weather Service (NWS).

Global Forecast System (GEFS)

A weather model created by the National Centers for Environmental Prediction (NCEP) that generates 21 separate forecasts to address underlying uncertainties in the input data.

High Resolution Rapid Refresh (HRRR)

3-km resolution hourly updated, cloud-resolving, convection-allowing atmospheric model.

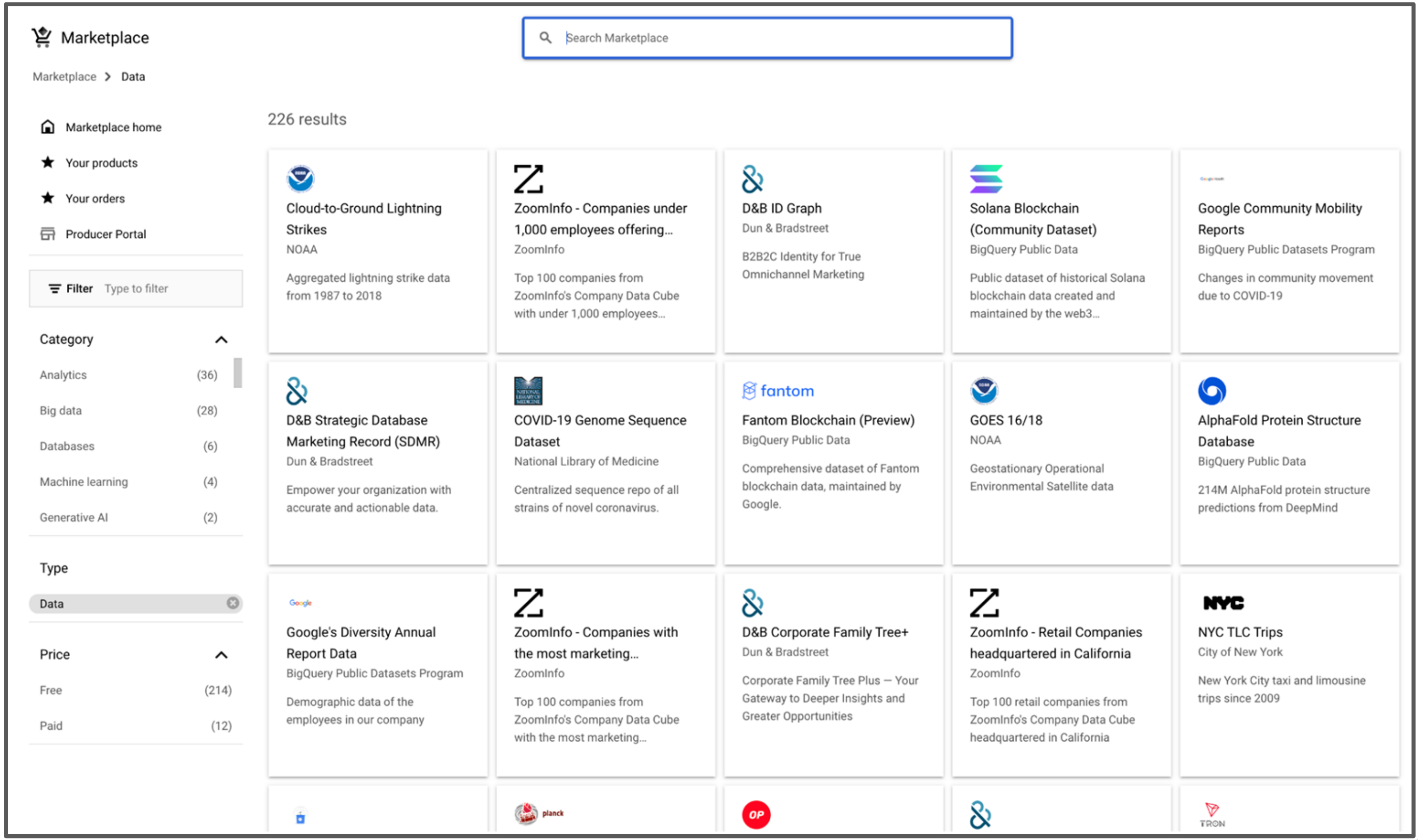
Global Historical Climatology Network (GHCN)

Integrated database of climate summaries from land surface stations across the globe.

02

Accessing and Using Public Datasets

Google Cloud Datasets Marketplace



Marketplace

Search Marketplace

Marketplace > Data

- Marketplace home
- Your products
- Your orders
- Producer Portal

Filter Type to filter

Category

Analytics	(36)
Big data	(28)
Databases	(6)
Machine learning	(4)
Generative AI	(2)

Type

Data

Price

Free	(214)
Paid	(12)

226 results

Cloud-to-Ground Lightning Strikes
NOAA
Aggregated lightning strike data from 1987 to 2018

ZoomInfo - Companies under 1,000 employees offering...
ZoomInfo
Top 100 companies from ZoomInfo's Company Data Cube with under 1,000 employees...

D&B ID Graph
Dun & Bradstreet
B2B2C Identity for True Omnichannel Marketing

Solana Blockchain (Community Dataset)
BigQuery Public Data
Public dataset of historical Solana blockchain data created and maintained by the web3...

Google Community Mobility Reports
BigQuery Public Datasets Program
Changes in community movement due to COVID-19

D&B Strategic Database Marketing Record (SDMR)
Dun & Bradstreet
Empower your organization with accurate and actionable data.

COVID-19 Genome Sequence Dataset
National Library of Medicine
Centralized sequence repo of all strains of novel coronavirus.

Fantom Blockchain (Preview)
BigQuery Public Data
Comprehensive dataset of Fantom blockchain data, maintained by Google.

GOES 16/18
NOAA
Geostationary Operational Environmental Satellite data

AlphaFold Protein Structure Database
BigQuery Public Data
214M AlphaFold protein structure predictions from DeepMind

Google's Diversity Annual Report Data
BigQuery Public Datasets Program
Demographic data of the employees in our company

ZoomInfo - Companies with the most marketing...
ZoomInfo
Top 100 companies from ZoomInfo's Company Data Cube with the most marketing...

D&B Corporate Family Tree+
Dun & Bradstreet
Corporate Family Tree Plus — Your Gateway to Deeper Insights and Greater Opportunities

ZoomInfo - Retail Companies headquartered in California
ZoomInfo
Top 100 retail companies from ZoomInfo's Company Data Cube headquartered in California

NYC TLC Trips
City of New York
New York City taxi and limousine trips since 2009

Where are public datasets stored?

Google Cloud Storage is a managed service for storing unstructured data.

Buckets contain objects (a.k.a. files and folders) that contain the data and how they're organized.

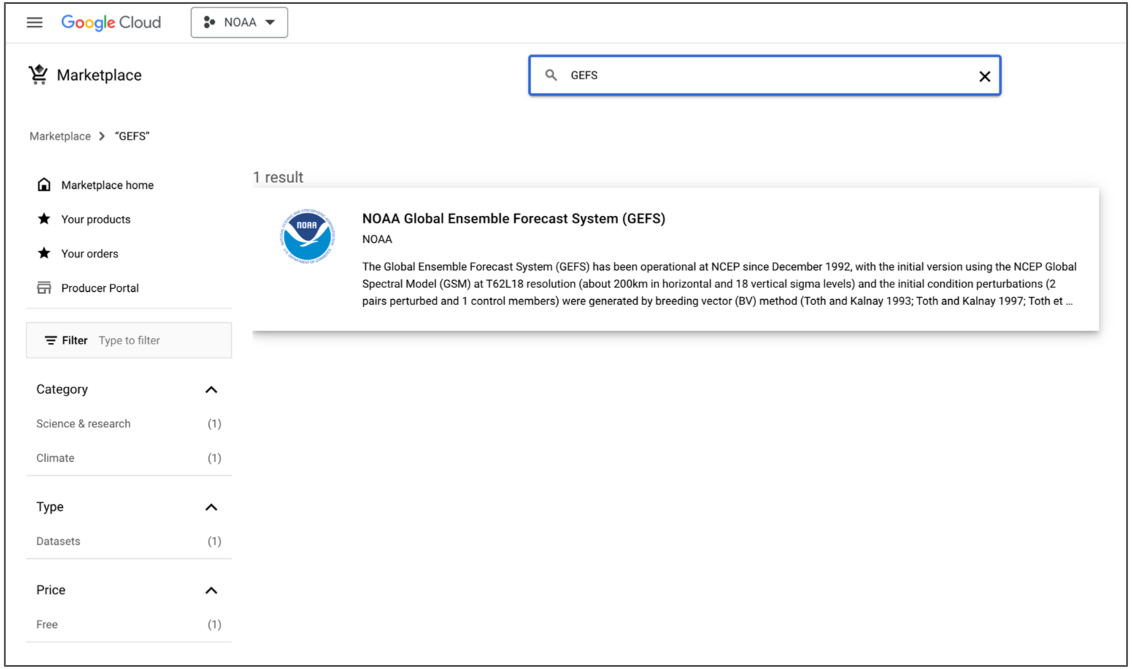
There are more than 80 buckets containing public datasets in various file formats.

BigQuery is Google's fully managed, serverless data warehouse for structured data.

It supports querying using a dialect of SQL.

There are more than 300 public BigQuery datasets spanning thousands of tables.

GEFS data in the GCP Marketplace



Category	
Science & research	(1)
Climate	(1)

Type	
Datasets	(1)

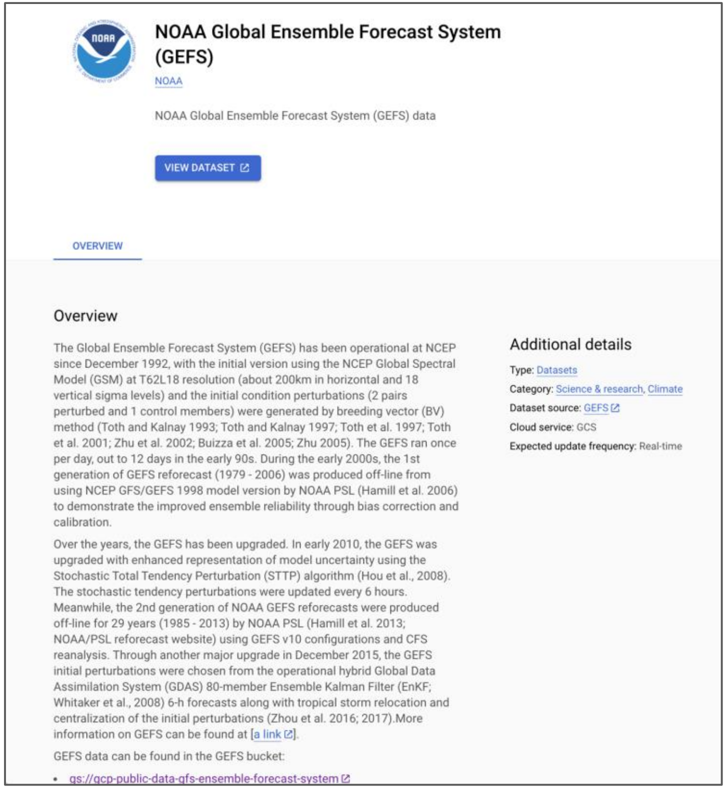
Price	
Free	(1)

1 result

NOAA Global Ensemble Forecast System (GEFS)

NOAA

The Global Ensemble Forecast System (GEFS) has been operational at NCEP since December 1992, with the initial version using the NCEP Global Spectral Model (GSM) at T62L18 resolution (about 200km in horizontal and 18 vertical sigma levels) and the initial condition perturbations (2 pairs perturbed and 1 control members) were generated by breeding vector (BV) method (Toth and Kalnay 1993; Toth and Kalnay 1997; Toth et ...



NOAA Global Ensemble Forecast System (GEFS)

NOAA

NOAA Global Ensemble Forecast System (GEFS) data

Overview

The Global Ensemble Forecast System (GEFS) has been operational at NCEP since December 1992, with the initial version using the NCEP Global Spectral Model (GSM) at T62L18 resolution (about 200km in horizontal and 18 vertical sigma levels) and the initial condition perturbations (2 pairs perturbed and 1 control members) were generated by breeding vector (BV) method (Toth and Kalnay 1993; Toth and Kalnay 1997; Toth et al. 1997; Toth et al. 2001; Zhu et al. 2002; Buizza et al. 2005; Zhu 2005). The GEFS ran once per day, out to 12 days in the early 90s. During the early 2000s, the 1st generation of GEFS reforecast (1979 - 2006) was produced off-line from using NCEP GFS/GEFS 1998 model version by NOAA PSL (Hamill et al. 2006) to demonstrate the improved ensemble reliability through bias correction and calibration.

Over the years, the GEFS has been upgraded. In early 2010, the GEFS was upgraded with enhanced representation of model uncertainty using the Stochastic Total Tendency Perturbation (STTP) algorithm (Hou et al., 2008). The stochastic tendency perturbations were updated every 6 hours. Meanwhile, the 2nd generation of NOAA GEFS reforecasts were produced off-line for 29 years (1985 - 2013) by NOAA PSL (Hamill et al. 2013; NOAA/PSL reforecast website) using GEFS v10 configurations and CFS reanalysis. Through another major upgrade in December 2015, the GEFS initial perturbations were chosen from the operational hybrid Global Data Assimilation System (GDAS) 80-member Ensemble Kalman Filter (EnKF; Whitaker et al., 2008) 6-h forecasts along with tropical storm relocation and centralization of the initial perturbations (Zhou et al. 2016; 2017).More information on GEFS can be found at [a link].

GEFS data can be found in the GEFS bucket:

- gs://gcp-public-data-gfs-ensemble-forecast-system

Additional details

Type: Datasets
Category: Science & research, Climate
Dataset source: GEFS
Cloud service: GCS
Expected update frequency: Real-time

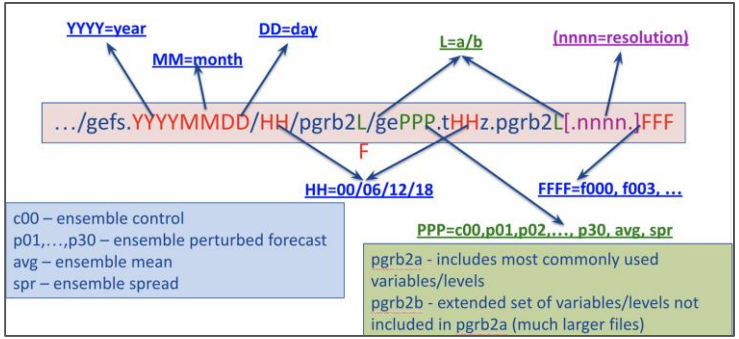
https://console.cloud.google.com/marketplace/product/noaa-public/gfs-ensemble-forecast-system

Access GEFS data using a browser

1. The Google Cloud Storage bucket that stores the data is gfs-ensemble-forecast-system
2. Using a web browser, access the root of the bucket with the following URI*
   https://console.cloud.google.com/storage/browser/gfs-ensemble-forecast-system
3. You can examine multiple levels of the bucket by appending the path to the URI above.

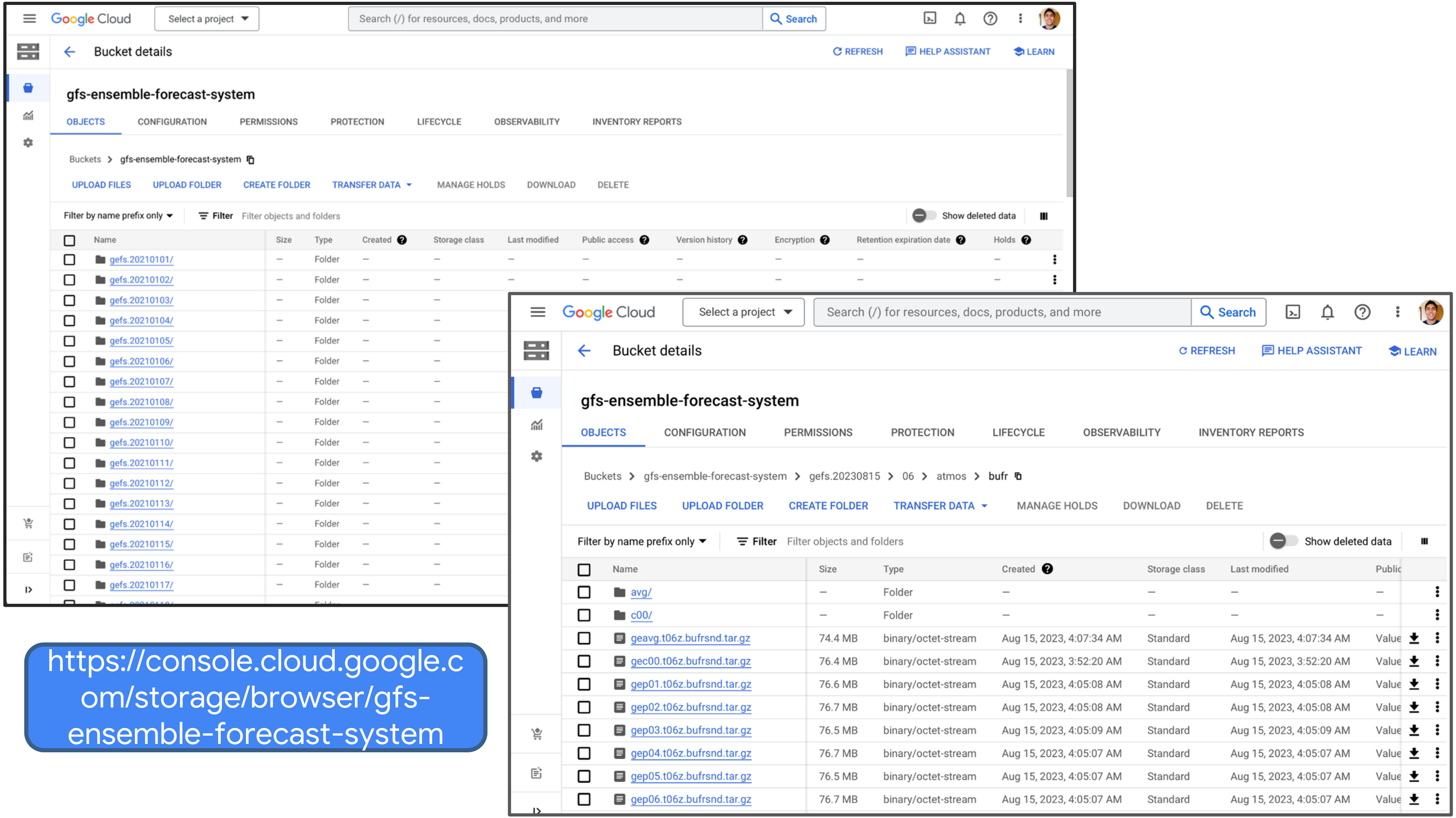
   For example, to access the path gefs.20230815/06/atmos/bufr, use the URI
   https://console.cloud.google.com/storage/browser/gfs-ensemble-forecast-system/gefs.20230815/06/atmos/bufr

*You will be asked to sign in if you are not currently signed in


YYYY=year
MM=month
DD=day
L=a/b
(nnnn=resolution)
…/gefs.YYYYMMDD/HH/pgrb2L/gePPP.tHHz.pgrb2L[.nnnn.]FFF
F
HH=00/06/12/18
FFFF=f000, f003, …
PPP=c00,p01,p02,…, p30, avg, spr

c00 – ensemble control
p01,…,p30 – ensemble perturbed forecast
avg – ensemble mean
spr – ensemble spread

pgrb2a - includes most commonly used variables/levels
pgrb2b - extended set of variables/levels not included in pgrb2a (much larger files)



Bucket details

gfs-ensemble-forecast-system

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE | OBSERVABILITY | INVENTORY REPORTS

Buckets > gfs-ensemble-forecast-system

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date	Holds
gefs.20210101/	—	Folder	—	—	—	—	—	—	—	—
gefs.20210102/	—	Folder	—	—	—	—	—	—	—	—
gefs.20210103/	—	Folder	—	—						
gefs.20210104/	—	Folder	—	—						
gefs.20210105/	—	Folder	—	—						
gefs.20210106/	—	Folder	—	—						
gefs.20210107/	—	Folder	—	—						
gefs.20210108/	—	Folder	—	—						
gefs.20210109/	—	Folder	—	—						
gefs.20210110/	—	Folder	—	—						
gefs.20210111/	—	Folder	—	—						
gefs.20210112/	—	Folder	—	—						
gefs.20210113/	—	Folder	—	—						
gefs.20210114/	—	Folder	—	—						
gefs.20210115/	—	Folder	—	—						
gefs.20210116/	—	Folder	—	—						
gefs.20210117/	—	Folder	—	—						

Bucket details

gfs-ensemble-forecast-system

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE | OBSERVABILITY | INVENTORY REPORTS

Buckets > gfs-ensemble-forecast-system > gefs.20230815 > 06 > atmos > bufr

Name	Size	Type	Created	Storage class	Last modified	Public
avg/	—	Folder	—	—	—	—
c00/	—	Folder	—	—	—	—
geavg.t06z.bufrsnd.tar.gz	74.4 MB	binary/octet-stream	Aug 15, 2023, 4:07:34 AM	Standard	Aug 15, 2023, 4:07:34 AM	Value
gec00.t06z.bufrsnd.tar.gz	76.4 MB	binary/octet-stream	Aug 15, 2023, 3:52:20 AM	Standard	Aug 15, 2023, 3:52:20 AM	Value
gep01.t06z.bufrsnd.tar.gz	76.6 MB	binary/octet-stream	Aug 15, 2023, 4:05:08 AM	Standard	Aug 15, 2023, 4:05:08 AM	Value
gep02.t06z.bufrsnd.tar.gz	76.7 MB	binary/octet-stream	Aug 15, 2023, 4:05:08 AM	Standard	Aug 15, 2023, 4:05:08 AM	Value
gep03.t06z.bufrsnd.tar.gz	76.5 MB	binary/octet-stream	Aug 15, 2023, 4:05:09 AM	Standard	Aug 15, 2023, 4:05:09 AM	Value
gep04.t06z.bufrsnd.tar.gz	76.7 MB	binary/octet-stream	Aug 15, 2023, 4:05:07 AM	Standard	Aug 15, 2023, 4:05:07 AM	Value
gep05.t06z.bufrsnd.tar.gz	76.5 MB	binary/octet-stream	Aug 15, 2023, 4:05:07 AM	Standard	Aug 15, 2023, 4:05:07 AM	Value
gep06.t06z.bufrsnd.tar.gz	76.7 MB	binary/octet-stream	Aug 15, 2023, 4:05:07 AM	Standard	Aug 15, 2023, 4:05:07 AM	Value

https://console.cloud.google.com/storage/browser/gfs-ensemble-forecast-system

Access GEFS data using the command line

gsutil is a Python application that lets you access Cloud Storage buckets and contents from the command line.

To list objects from the root of the bucket:

```
$ gsutil ls gs://gfs-ensemble-forecast-system

gs://gfs-ensemble-forecast-system/gefs.20210101/
gs://gfs-ensemble-forecast-system/gefs.20210102/
gs://gfs-ensemble-forecast-system/gefs.20210103/
gs://gfs-ensemble-forecast-system/gefs.20210104/
gs://gfs-ensemble-forecast-system/gefs.20210105/
gs://gfs-ensemble-forecast-system/gefs.20210106/
gs://gfs-ensemble-forecast-system/gefs.20210107/
gs://gfs-ensemble-forecast-system/gefs.20210108/
...
```

Using gcloud storage has a similar effect:

```
$ gcloud storage ls gs://gfs-ensemble-forecast-system
```

Access GEFS data using the command line

To copy an entire prefix (directory tree) and its contents to the current directory*

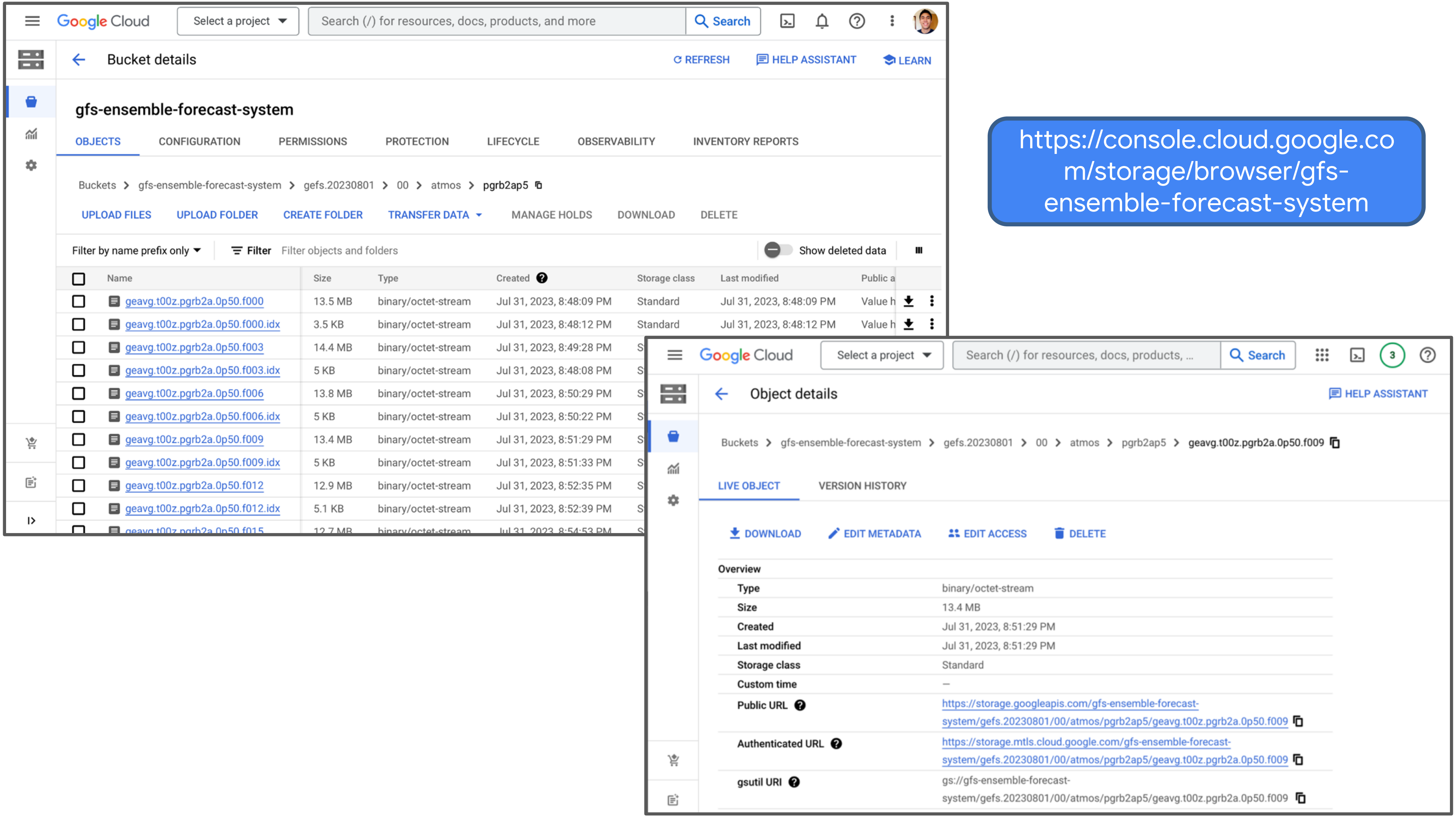
```
  $ gsutil -m cp gs://gfs-ensemble-forecast-system/gefs.20230812 .
```

*The -m flag enables multiprocessing to parallelize object downloads. Note that data for a single date (i.e. a `gefs.YYYYMMDD` folder) is more than 100 GB in size.

Again, using gcloud storage has a similar effect (without the -m flag):

```
  $ gcloud storage cp gs://gfs-ensemble-forecast-system/gefs.20230812 .
```

(For more info, see https://cloud.google.com/sdk/gcloud/reference/storage)



gfs-ensemble-forecast-system

Buckets > gfs-ensemble-forecast-system > gefs.20230801 > 00 > atmos > pgrb2ap5

	Name	Size	Type	Created	Storage class	Last modified	Public a
□	geavg.t00z.pgrb2a.0p50.f000	13.5 MB	binary/octet-stream	Jul 31, 2023, 8:48:09 PM	Standard	Jul 31, 2023, 8:48:09 PM	Value h
□	geavg.t00z.pgrb2a.0p50.f000.idx	3.5 KB	binary/octet-stream	Jul 31, 2023, 8:48:12 PM	Standard	Jul 31, 2023, 8:48:12 PM	Value h
□	geavg.t00z.pgrb2a.0p50.f003	14.4 MB	binary/octet-stream	Jul 31, 2023, 8:49:28 PM	S		
□	geavg.t00z.pgrb2a.0p50.f003.idx	5 KB	binary/octet-stream	Jul 31, 2023, 8:48:08 PM	S		
□	geavg.t00z.pgrb2a.0p50.f006	13.8 MB	binary/octet-stream	Jul 31, 2023, 8:50:29 PM	S		
□	geavg.t00z.pgrb2a.0p50.f006.idx	5 KB	binary/octet-stream	Jul 31, 2023, 8:50:22 PM	S		
□	geavg.t00z.pgrb2a.0p50.f009	13.4 MB	binary/octet-stream	Jul 31, 2023, 8:51:29 PM	S		
□	geavg.t00z.pgrb2a.0p50.f009.idx	5 KB	binary/octet-stream	Jul 31, 2023, 8:51:33 PM	S		
□	geavg.t00z.pgrb2a.0p50.f012	12.9 MB	binary/octet-stream	Jul 31, 2023, 8:52:35 PM	S		
□	geavg.t00z.pgrb2a.0p50.f012.idx	5.1 KB	binary/octet-stream	Jul 31, 2023, 8:52:39 PM	S		
□	geavg.t00z.pgrb2a.0p50.f015	12.7 MB	binary/octet-stream	Jul 31, 2023, 8:54:53 PM	S		

Buckets > gfs-ensemble-forecast-system > gefs.20230801 > 00 > atmos > pgrb2ap5 > geavg.t00z.pgrb2a.0p50.f009

Overview

Type	binary/octet-stream
Size	13.4 MB
Created	Jul 31, 2023, 8:51:29 PM
Last modified	Jul 31, 2023, 8:51:29 PM
Storage class	Standard
Custom time	—
Public URL	https://storage.googleapis.com/gfs-ensemble-forecast-system/gefs.20230801/00/atmos/pgrb2ap5/geavg.t00z.pgrb2a.0p50.f009
Authenticated URL	https://storage.mtls.cloud.google.com/gfs-ensemble-forecast-system/gefs.20230801/00/atmos/pgrb2ap5/geavg.t00z.pgrb2a.0p50.f009
gsutil URI	gs://gfs-ensemble-forecast-system/gefs.20230801/00/atmos/pgrb2ap5/geavg.t00z.pgrb2a.0p50.f009

https://console.cloud.google.com/storage/browser/gfs-ensemble-forecast-system

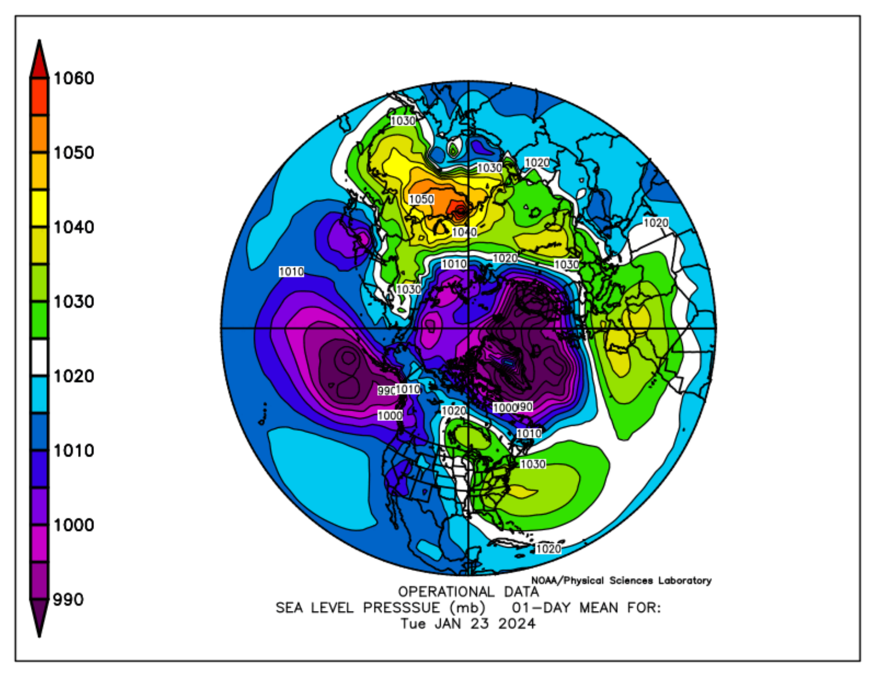
03

Use Cases and Journeys

Weather Satellites: An Invaluable Resource

- Nearly unlimited use cases. Weather observations for agriculture, transportation, finance, and energy sectors
- Climate change monitoring
- Disaster/risk management
  - Wildfires
  - Extreme Floods
  - Hurricanes, Tropical Storms, and Extreme Weather

Global Weather Models: Another Invaluable Resource

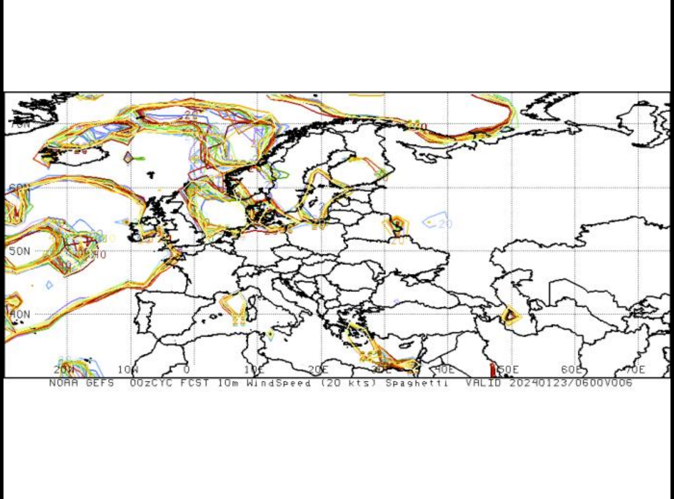


NOAA/Physical Sciences Laboratory

OPERATIONAL DATA
SEA LEVEL PRESSSUE (mb) 01–DAY MEAN FOR:
Tue JAN 23 2024

- Global weather models available in high resolution 4x per day for:
  - Business Analytics
  - Operational Forecast Needs
  - ML training and validation

- Global weather model ensembles available
  - Use 30 perturbed + 1 control forecast to increase your certainty in how much uncertainty a model has!



NOAA GEFS 00zCYC FCST 10m WindSpeed (20 kts) Spaghetti VALID 20240123/0600V006

Quick code example for JPSS ATMS data on GCP

Try this on your own Colab!

Setup

Setup your environment for all the tools you will need to accomplish your task.

```
!pip install -q zarr xarray[complete] fsspec aiohttp requests gcsfs cartopy

from google.colab import auth
from google.cloud import storage

from datetime import datetime
import xarray as xr
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import fsspec
import gcsfs

auth.authenticate_user()
fs = gcsfs.GCSFileSystem(project='something')
```

Instantiate client and fetch blobs

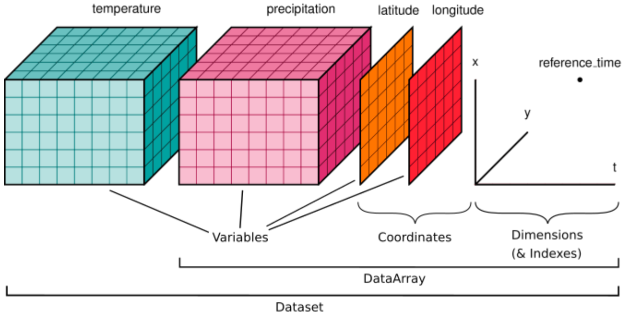
Start up your data set and find the data you want to work with. In this case we are working with JPSS MIRS data.

```
# Instantiates a client
storage_client = storage.Client()

# The bucket name for the JPSS VIIRS data
bucket_name = "noaa-nesdis-n20"

def list_blobs(bucket_name, prefix, delimiter=None):
    """Lists all the blobs in the bucket."""

    storage_client = storage.Client()
    blobs = storage_client.list_blobs(bucket_name,
prefix=prefix, delimiter=delimiter)
    # Note: The call returns a response only when the
iterator is consumed.
    results = []
    for blob in blobs:
        # print(blob.name)
        results.append(blob.name)
    return results

results = list_blobs(bucket_name,
"NPR_MIRS_IMG/2023/09/15", None)

# Alternative fetching method
# results = !gsutil ls 'gs://noaa-nesdis-
n20/NPR_MIRS_IMG/2023/09/15/'

# Get rid of 33 min data for this example
results = [results[i] for i in range(len(results)) if
results[i].find('33min')<0]
print('Number of files: ', len(results))
```

Identify the files we want to work with…

Isolate the files you want to work with and add some additional parsed data.

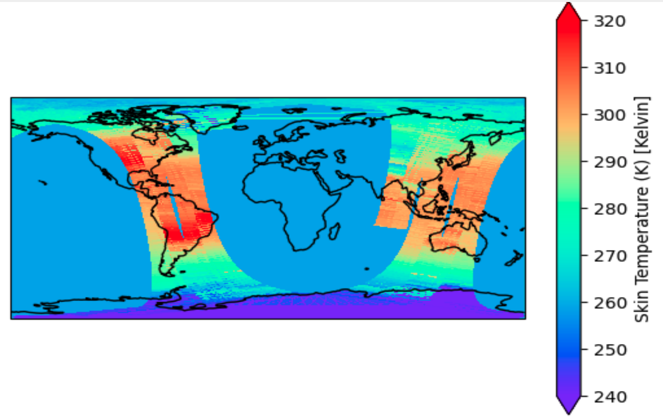
```
def parse_dates(string_names):
  string_names = string_names[48:62]
  year = int(string_names[0:4])
  month = int(string_names[4:6])
  day = int(string_names[6:8])
  hour = int(string_names[8:10])
  minute = int(string_names[10:12])
  seconds= int(string_names[12:14])
  string_dt = datetime(year, month, day, hour, minute,
seconds)
  return string_dt

# Create a dataframe and add datetime field from filenames
dfr = pd.DataFrame(results, columns=['Files'])
dfr['Date'] = dfr.Files.apply(parse_dates)

#Fetch data from an important time
lets_get = dfr[(dfr.Date >= "2023-09-15 15:30:00") &
               (dfr.Date < "2023-09-15 23:30:00")]

# Get Filenames
Lets_get = lets_get.Files.to_list()
print('Filtered to:', len(lets_get))

# Show first 5 files
lets_get[0:5]
```



Open and process each file

Process netcdf files and plot using xarray and matplotlib.

```
datasets = []
for file in lets_get:
  data_path = 'gs://' + bucket_name + '/' + file
  ds3 = xr.open_dataset(fs.open(data_path), engine='h5netcdf')

  # Reduce data set to Skin temperature data
  datasets.append(ds3['TSkin'])

# Concatenate all the data to make one image from all data sources
combined = xr.concat(datasets, dim='Field_of_view')

from cartopy import config
import matplotlib.pyplot as plt
import cartopy.crs as ccrs

ax = plt.axes(projection=ccrs.PlateCarree())
combined.plot.pcolormesh(x='Longitude', y='Latitude',
cmap='rainbow', transform=ccrs.PlateCarree())

ax.coastlines()
plt.show()
```



Quick code example for JPSS VIIRS Longwave IR data on GCP

Try this on your own Colab!
https://colab.research.google.com

Imports and Setup

Setup your kernel for all the tools you will need to accomplish these processing tasks.

```
!pip install -q zarr xarray[complete] fsspec aiohttp requests gcsfs cartopy

import h5py
import gcsfs
import matplotlib.pyplot as plt
from google.colab import auth
from google.cloud import storage
from datetime import datetime
import pandas as pd
import numpy as np
from cartopy import config
import matplotlib.pyplot as plt
import cartopy.crs as ccrs

auth.authenticate_user()
```

Setup variables for data processing steps

Making some useful variables to limit data processing to reasonable size for processing on basic Colab.

```
bucket_name = "noaa-nesdis-n21"
target_data = "VIIRS-I4-IMG-EDR"
target_data_geo = "VIIRS-IMG-GTM-EDR-GEO"

year = '2023'
month = '09'
day = '15'
start_hour = '17'
start_minute = '13'
end_hour = '17'
end_minute = '17'

start_limiter = datetime(int(year), int(month), int(day),
int(start_hour), int(start_minute), 0)
end_limiter = datetime(int(year), int(month), int(day),
int(end_hour), int(end_minute), 0)
fs = gcsfs.GCSFileSystem(anon=True)
```

Instantiate client to fetch IR data blobs

Find all the relevant data from the IR sensor. In this case we are working with JPSS VIIRS IR data so we are focused on the I4 band and during our time range.

```
# Instantiates a client
storage_client = storage.Client()

def list_blobs(bucket_name, prefix, delimiter=None):
    """Lists all the blobs in the bucket."""

    storage_client = storage.Client()
    blobs = storage_client.list_blobs(bucket_name, prefix=prefix, delimiter=delimiter)
    # Note: The call returns a response only when the iterator is consumed.
    results = []
    for blob in blobs:
        # print(blob.name)
        results.append(blob.name)
    return results

results = list_blobs(bucket_name, f"{target_data}/{year}/{month}/{day}/", None)

print('Number of files: ', len(results))

def parse_dates(s):
  s = s[38:56]
  year = int(s[1:5])
  month = int(s[5:7])
  day = int(s[7:9])
  hour = int(s[11:13])
  minute = int(s[13:15])
  seconds= int(s[15:17])
  string_dt = datetime(year, month, day, hour, minute, seconds)
  return string_dt

dfr = pd.DataFrame(results, columns=['Files'])
dfr['Date'] = dfr.Files.apply(parse_dates)

lets_get = dfr[(dfr.Date >= start_limiter) & (dfr.Date < end_limiter)].Files.to_list()
print('Filtered to:', len(lets_get))
lets_get[0:5]
```

Instantiate client to fetch geo blobs

Start up your data set and find the data you want to work with. In this case we are working with grabbing GEO data.

```
# Instantiates a client
storage_client = storage.Client()

def list_blobs(bucket_name, prefix, delimiter=None):
    """Lists all the blobs in the bucket."""

    storage_client = storage.Client()
    blobs = storage_client.list_blobs(bucket_name, prefix=prefix, delimiter=delimiter)
    # Note: The call returns a response only when the iterator is consumed.
    results = []
    for blob in blobs:
        # print(blob.name)
        results.append(blob.name)
    return results

results_geo = list_blobs(bucket_name, f"{target_data_geo}/{year}/{month}/{day}/", None)

print('Number of files: ', len(results))

def parse_dates(s):
  s = s[43:61]
  year = int(s[1:5])
  month = int(s[5:7])
  day = int(s[7:9])
  hour = int(s[11:13])
  minute = int(s[13:15])
  seconds= int(s[15:17])
  string_dt = datetime(year, month, day, hour, minute, seconds)
  return string_dt

dfr_geo = pd.DataFrame(results_geo, columns=['Files'])
dfr_geo['Date'] = dfr_geo.Files.apply(parse_dates)

lets_get_geo = dfr_geo[(dfr_geo.Date >= start_limiter) & (dfr_geo.Date <
end_limiter)].Files.to_list()
print('Filtered to:', len(lets_get_geo))
lets_get_geo[0:5]
```

Process all the data you need for your project.

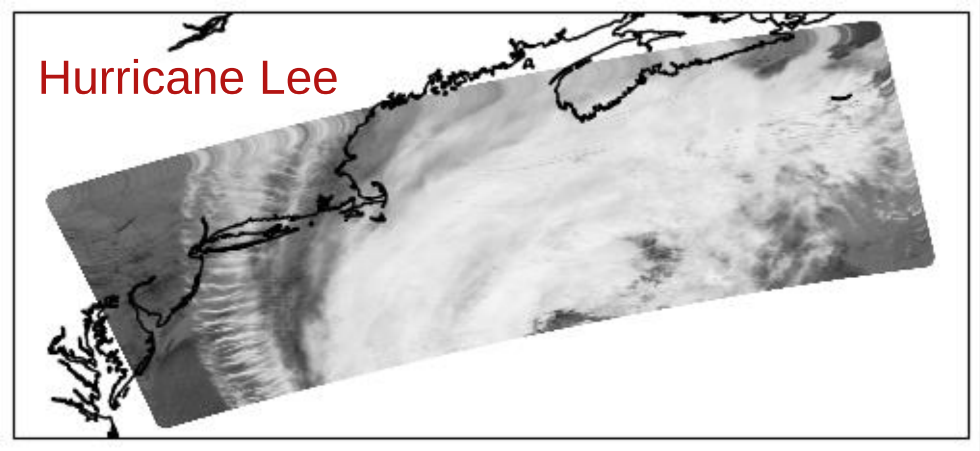
Process data to numpy arrays (online) and prep for visualization.

```
agg_data_lat = []
agg_data_lon = []

# One file is ALOT of data
for file in lets_get_geo[1:2]:
  r = 'gs://' + bucket_name + '/' + file
  print(r)
  f = h5py.File(fs.open(r), 'r')
  latitude = f['All_Data']['VIIRS-IMG-GTM-EDR-GEO_All']['Latitude'][:]
  print(np.array(latitude).shape)
  longitude = f['All_Data']['VIIRS-IMG-GTM-EDR-GEO_All']['Longitude'][:]
  print(np.array(longitude).shape)

  # If processing multiple files at once
  for row in range(latitude.shape[0]):
    agg_data_lat.append(latitude[row])
  for row in range(longitude.shape[0]):
    agg_data_lon.append(longitude[row])

agg_data = []
counter = 0

for file in lets_get[1:2]:
  r = 'gs://' + bucket_name + '/' + file
  print(r)
  f = h5py.File(fs.open(r), 'r')
  if counter < 1:
    products = f['All_Data']['VIIRS-I4-IMG-EDR_All']

  arr = f['All_Data']['VIIRS-I4-IMG-EDR_All']['BrightnessTemperature'][:]
  print(np.array(arr).shape)
  for row in range(0, arr.shape[0]):
    agg_data.append(arr[row])

 longitude[longitude<-900] = np.nan
 latitude[latitude<-900] = np.nan
```

Visualize Numpy Arrays

Plot numpy arrays using matplotlib and cartopy.

```
from cartopy import config
import matplotlib.pyplot as plt
import cartopy.crs as ccrs

ax = plt.axes(projection=ccrs.PlateCarree())
plt.scatter(longitude[:, 2500:6500], latitude[:, 2500:6500],
c=arr[:, 2500:6500], cmap='Greys',
transform=ccrs.PlateCarree())

ax.coastlines()
plt.show()
```



Thank you.

Questions? Email us at:
cloud-public-dataset-conferences@google.com

Google Cloud