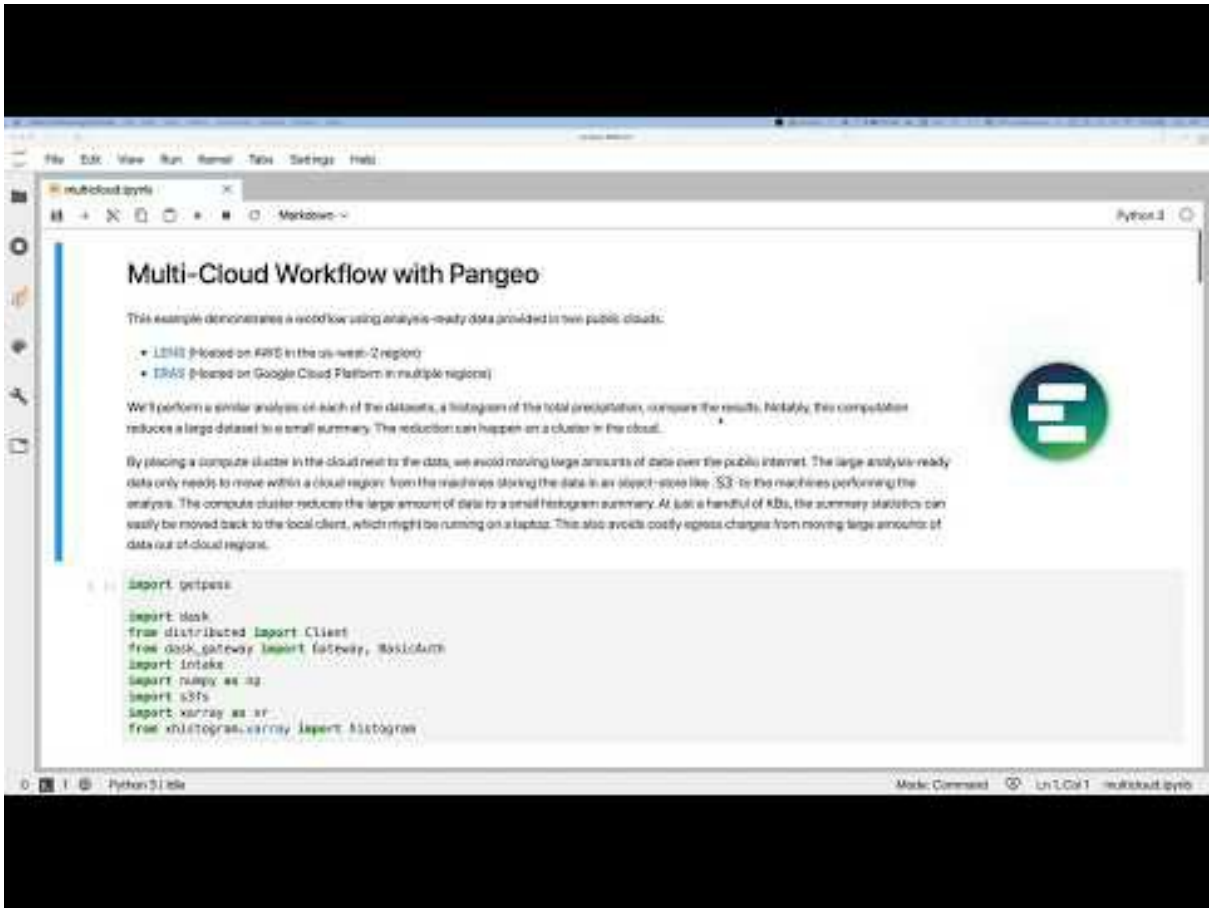
Multi-Cloud workflows with Pangeo and Dask Gateway

Tom Augspurger, Martin Durant, Ryan Abernathey, Joe Hamman

As more analysis-ready datasets are provided on the cloud, we need to consider how researchers access data. To maximize performance and minimize costs, we move the analysis to the data. This notebook demonstrates a Pangeo deployment connected to multiple Dask Gateways to enable analysis, regardless of where the data is stored. Public clouds are partitioned into regions, a geographic location with a cluster of data centers. A dataset like the National Water Model Short-Range Forecast is provided in a single region of some cloud provider (e.g. AWS's us-east-1). To analyze that dataset efficiently, we do the analysis in the same region as the dataset. That's especially true for very large datasets. Making local "dark replicas" of the datasets is slow and expensive. In this notebook we demonstrate a few open source tools to compute "close" to cloud data. We use Intake as a data catalog, to discover the datasets we have available and load them as an xarray Dataset. With xarray, we're able to write the necessary transformations, filtering, and reductions that compose our analysis. To process the large amounts of data in parallel, we use Dask. Behind the scenes, we've configured this Pangeo deployment with multiple Dask Gateways, which provide a secure, multi-tenant server for managing Dask clusters. Each Gateway is provisioned with the necessary permissions to access the data. By placing compute (the Dask workers) in the same region as the dataset, we achieve the highest performance: these worker machines are physically close to the machines storing the data and have the highest bandwidth. We minimize cost by avoiding egress costs: fees charged to the data provider when data leaves a cloud region.

Demonstration using Pangeo deployments to work with datasets provided in
multiple cloud regions.

This screencast (https://www.youtube.com/watch?v=IeKjLiUqpT4) demonstrates the notebook.



(https://www.youtube.com/watch?v=IeKjLiUqpT4)

Rendered Notebook: https://nbviewer.jupyter.org/github/pangeo-data/multicloud-demo/blob/master/multicloud.ipynb

Multi-Cloud Workflow with Pangeo

This example demonstrates a workflow using analysis-ready data provided in two public clouds.

- LENS (Hosted on AWS in the us-west-2 region)
- ERA5 (Hosted on Google Cloud Platform in multiple regions)

We'll perform a similar analysis on each of the datasets, a histogram of the total precipitation, compare the results. Notably, this computation reduces a large dataset to a small summary. The reduction can happen on a cluster in the cloud.

By placing a compute cluster in the cloud next to the data, we avoid moving large amounts of data over the public internet. The large analysis-ready data only needs to move within a cloud region: from the machines storing the data in an object-store like `S3` to the machines performing the analysis. The compute cluster reduces the large amount of data to a small histogram summary. At just a handful of KBs, the summary statistics can easily be moved back to the local client, which might be running on a laptop. This also avoids costly egress charges from moving large amounts of data out of cloud regions.

In [1]:
```
import getpass

import dask
from distributed import Client
from dask_gateway import Gateway, BasicAuth
import intake
import numpy as np
import s3fs
import xarray as xr
from xhistogram.xarray import histogram
```

Create Dask Clusters

We've deployed Dask Gateway on two Kubernetes clusters, one in AWS and one in GCP. We'll use these to create Dask clusters in the same cloud region as the data. We'll connect to both of them from the same interactive notebook session.

In [2]:
```
password = getpass.getpass()
auth = BasicAuth("pangeo", password)
```

In [3]:

```
# Create a Dask Cluster on AWS
aws_gateway = Gateway(
    "http://a00670d37945911eab47102a1da71b1b-524946043.us-west-2.elb.amazonaws.com",
    auth=auth,
)
aws = aws_gateway.new_cluster()
aws_client = Client(aws, set_as_default=False)
aws_client
```

Out[3]:

Client

Scheduler: gateway://a00670d37945911eab47102a1da71b1b-524946043.us-west-2.elb.amazonaws.com:80/dask-gateway.ff367abfd96c4465a0782661a254e589
Dashboard: http://a00670d37945911eab47102a1da71b1b-524946043.us-west-2.elb.amazonaws.com/clusters/dask-gateway.ff367abfd96c4465a0782661a254e589/status

Cluster

Workers: 0
Cores: 0
Memory: 0 B

In [4]:

```
# Create a Dask Cluster on GCP
gcp_gateway = Gateway(
    "http://34.72.56.89",
    auth=auth,
)
gcp = gcp_gateway.new_cluster()
gcp_client = Client(gcp, set_as_default=False)
gcp_client
```

Out[4]:

Client

Scheduler: gateway://34.72.56.89:80/dask-gateway.02ab011eaa054434916da9d3e3405c6c
Dashboard: http://34.72.56.89/clusters/dask-gateway.02ab011eaa054434916da9d3e3405c6c/status

Cluster

Workers: 0
Cores: 0
Memory: 0 B

We'll enable adaptive mode on each of the Dask clusters. Workers will be added and removed as needed by the current level of computation.

In [5]:

```
aws.adapt(minimum=1, maximum=200)
gcp.adapt(minimum=1, maximum=200)
```

ERA5 on Google Cloud Storage

We'll use `intake` and pangeo's data catalog to discover the dataset.

In [6]:
```
cat = intake.open_catalog(
    "https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/master.yaml"
)
cat
```

Out[6]:
```
<Intake catalog: master>
```

The next cell loads the *metadata* as an xarray dataset. No large amount of data is read or transfered here. It will be loaded on-demand when we ask for a concrete result later.

In [7]:
```
era5 = cat.atmosphere.era5_hourly_reanalysis_single_levels_sa(
    storage_options={"requester_pays": False, "token": "anon"}
).to_dask()
era5
```

Out[7]: xarray.Dataset

▸ Dimensions: (**latitude**: 721, **longitude**: 1440, **time**: 350640)

▾ Coordinates:

latitude	(latitude)	float32	90.0 89.75 89.5 ... -89.75 -90.0
longitude	(longitude)	float32	0.0 0.25 0.5 ... 359.5 359.75
time	(time)	datetime64[ns]	1979-01-01 ... 2018-12-31T23:00:00

▸ Data variables: (17)