Data Everywhere: Using and Sharing Scientific Data with Pelican

Andrew Owen, PhD

Research Computing Facilitator

Center for High Throughput Computing

University of Wisconsin - Madison

Setup

All instructions, materials are in Github repository

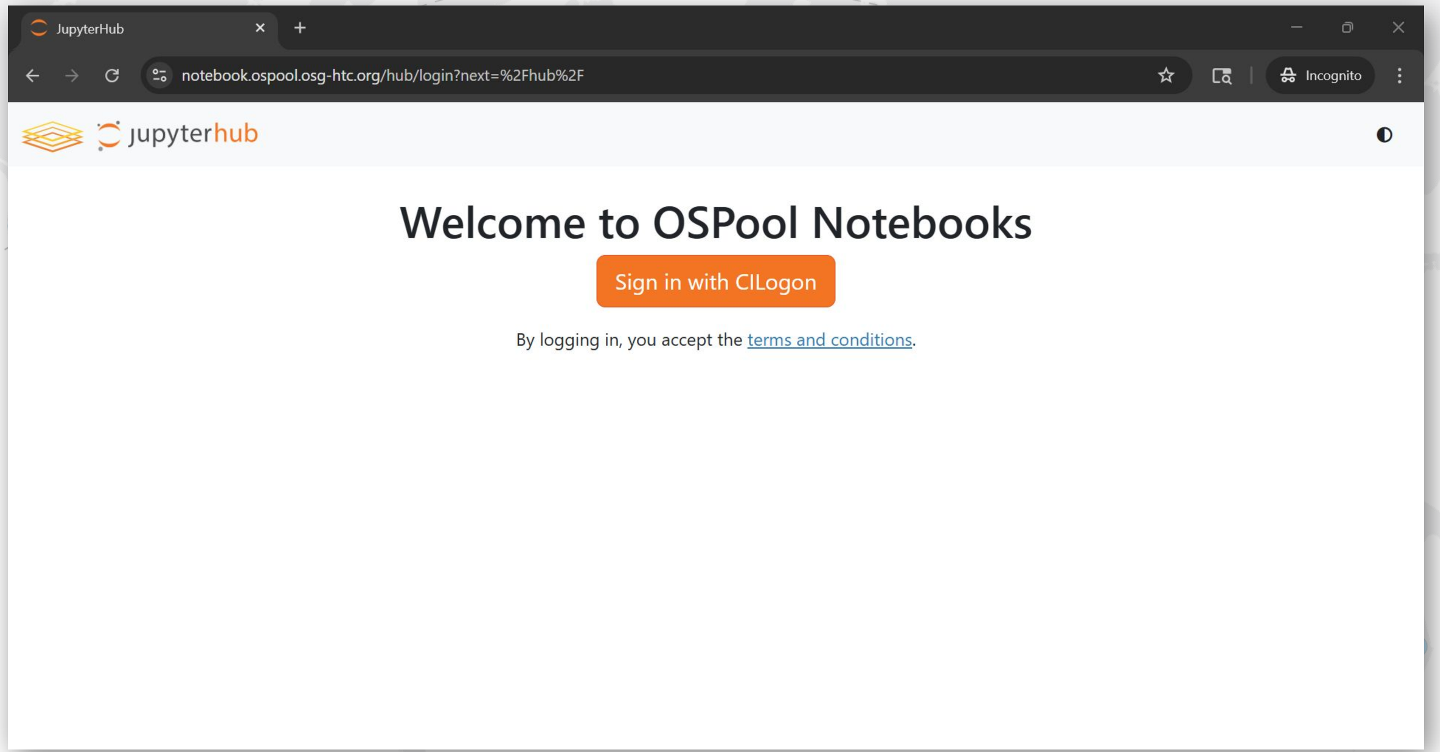
github.com/pelicanplatform/training-client

Quickstart

1. Sign in to a Guest notebook at notebook.ospool.osg-htc.org using your institution ID.
2. Open a "Terminal" tab in the Jupyter Notebook

Quickstart

notebook.ospool.osg-htc.org



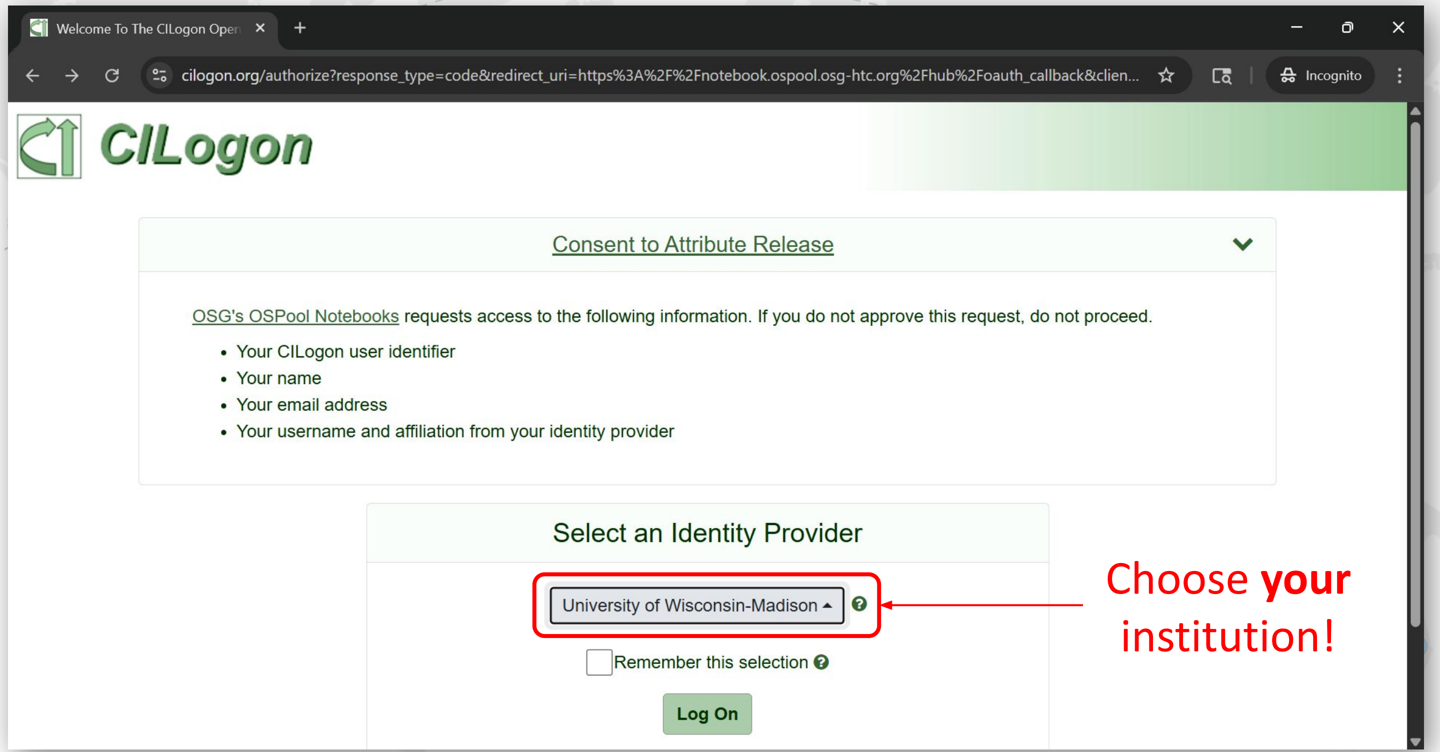
Welcome to OSPool Notebooks

Sign in with CILogon

By logging in, you accept the terms and conditions.

Quickstart

notebook.ospool.osg-htc.org



Welcome To The CILogon Open
cilogon.org/authorize?response_type=code&redirect_uri=https%3A%2F%2Fnotebook.ospool.osg-htc.org%2Fhub%2Foauth_callback&clien...
Incognito
CILogon

Consent to Attribute Release

OSG's OSPool Notebooks requests access to the following information. If you do not approve this request, do not proceed.

- Your CILogon user identifier
- Your name
- Your email address
- Your username and affiliation from your identity provider

Select an Identity Provider

University of Wisconsin-Madison

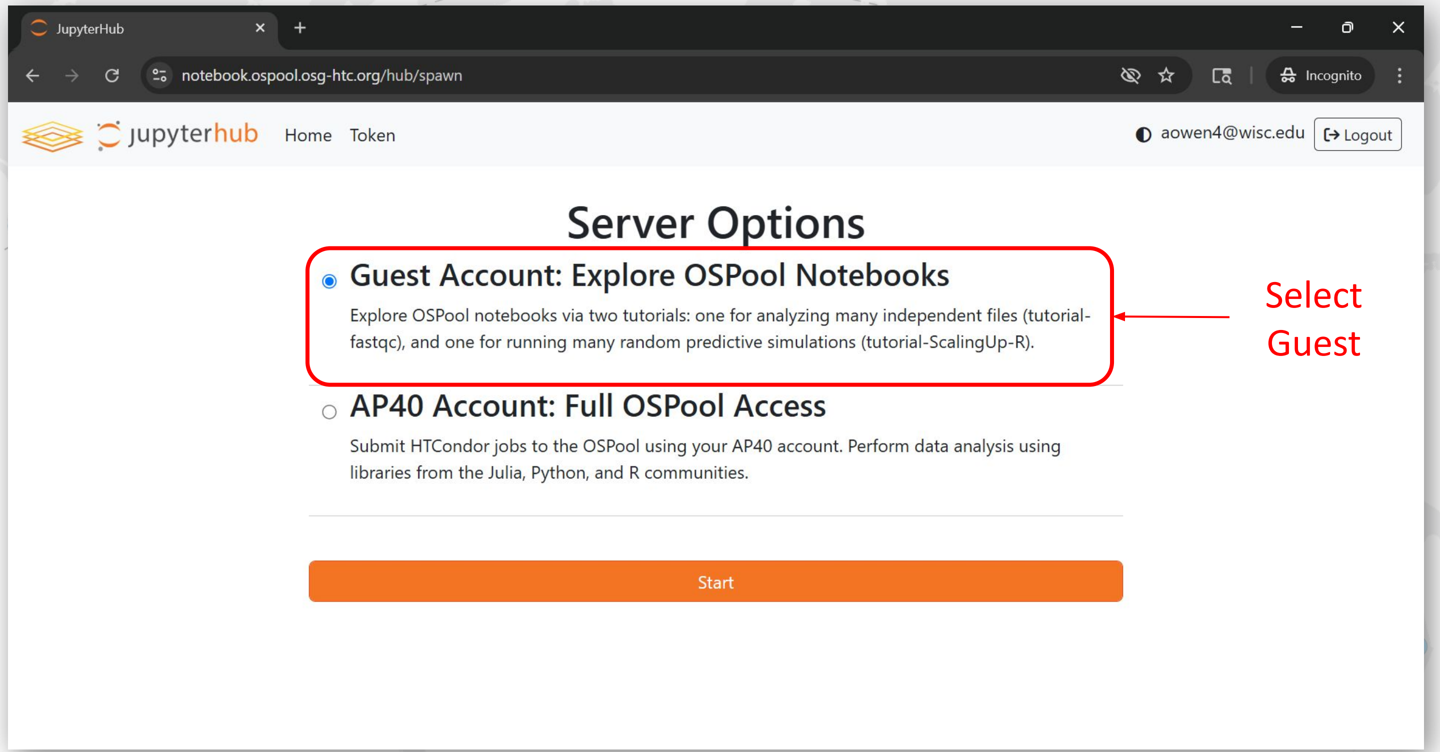
Remember this selection

Log On

Choose **your** institution!

Quickstart

notebook.ospool.osg-htc.org



Server Options

[✓ Guest Account: Explore OSPool Notebooks]

Explore OSPool notebooks via two tutorials: one for analyzing many independent files (tutorial-fastqc), and one for running many random predictive simulations (tutorial-ScalingUp-R).

AP40 Account: Full OSPool Access

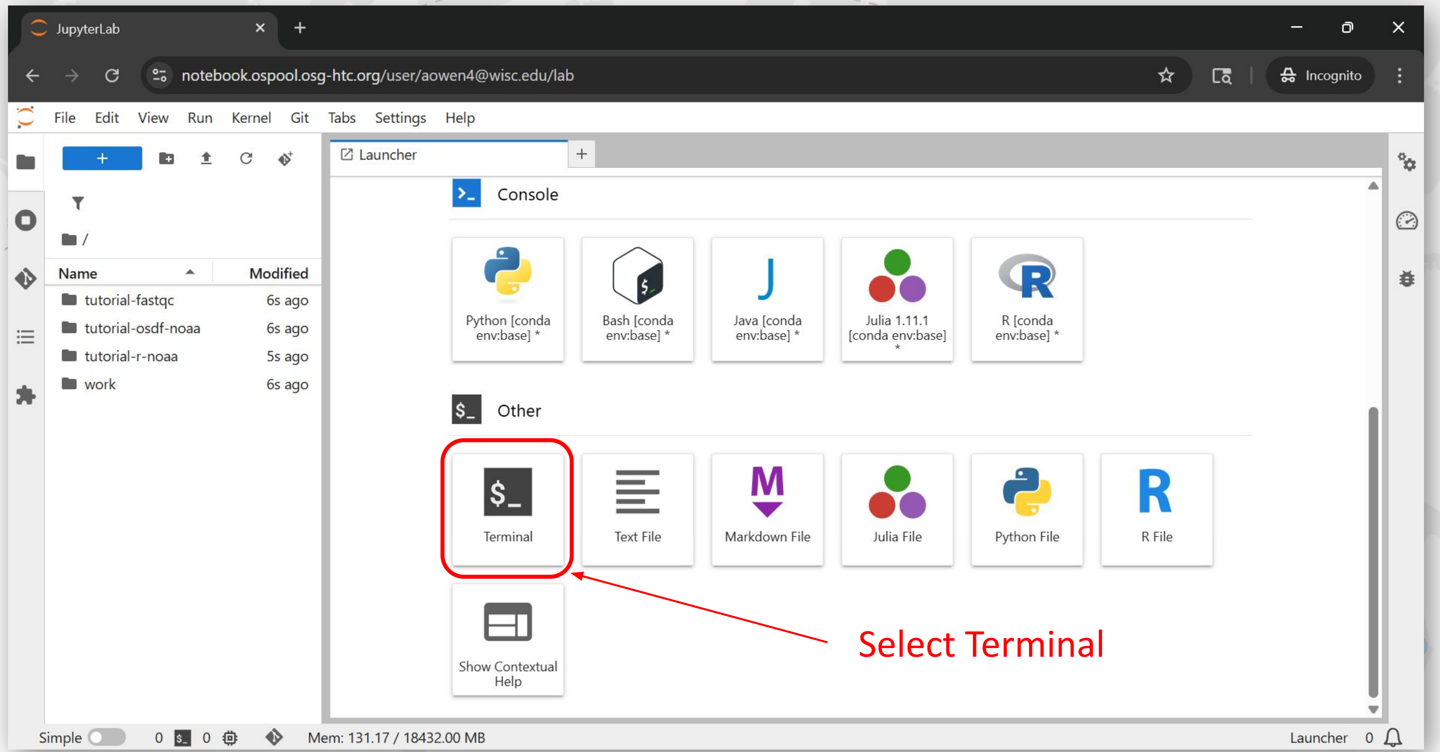
Submit HTCondor jobs to the OSPool using your AP40 account. Perform data analysis using libraries from the Julia, Python, and R communities.

Start

Select Guest

Quickstart

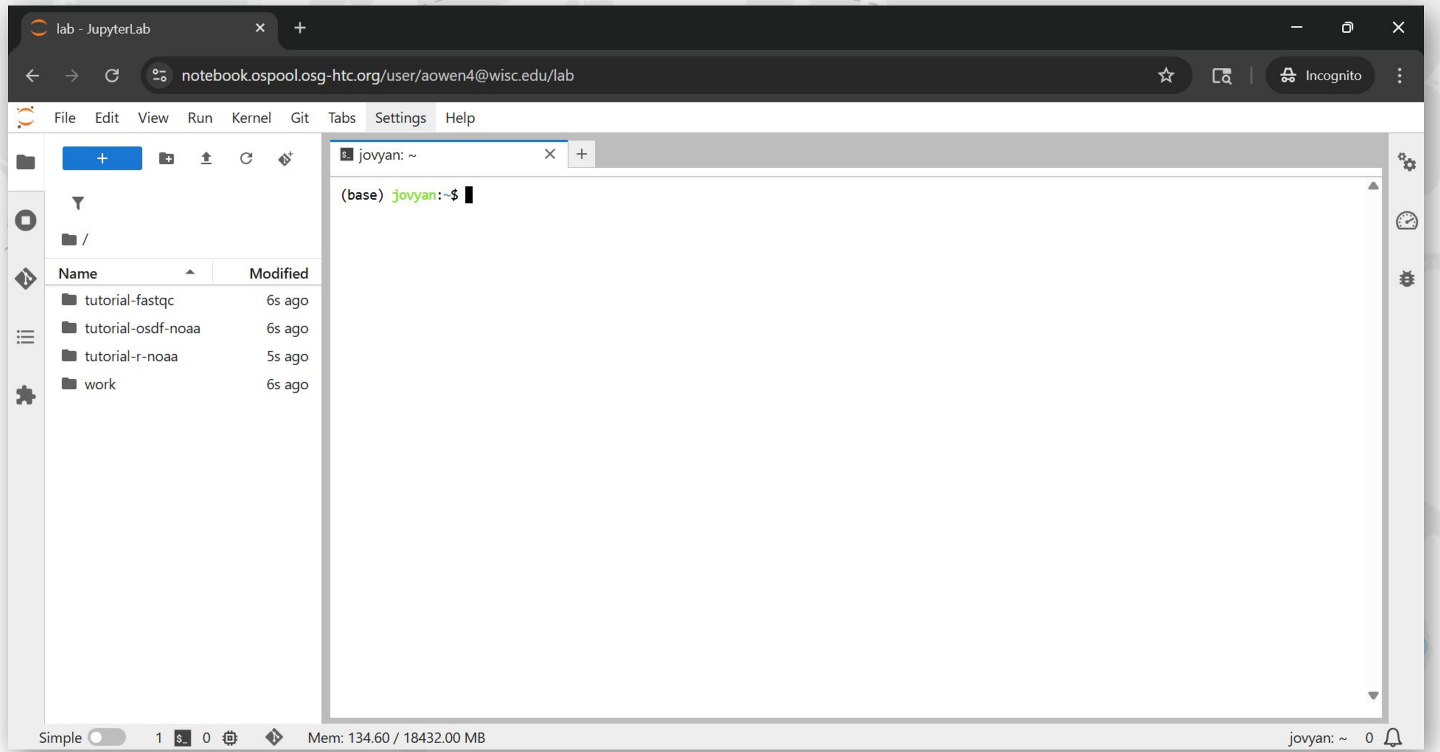
notebook.ospool.osg-htc.org



Select Terminal

Quickstart

notebook.ospool.osg-htc.org


lab - JupyterLab
notebook.ospool.osg-htc.org/user/aowen4@wisc.edu/lab
Incognito
File Edit View Run Kernel Git Tabs Settings Help
jovyan: ~
Simple
Mem: 134.60 / 18432.00 MB
jovyan: ~

Name	Modified
tutorial-fastqc	6s ago
tutorial-osdf-noaa	6s ago
tutorial-r-noaa	5s ago
work	6s ago

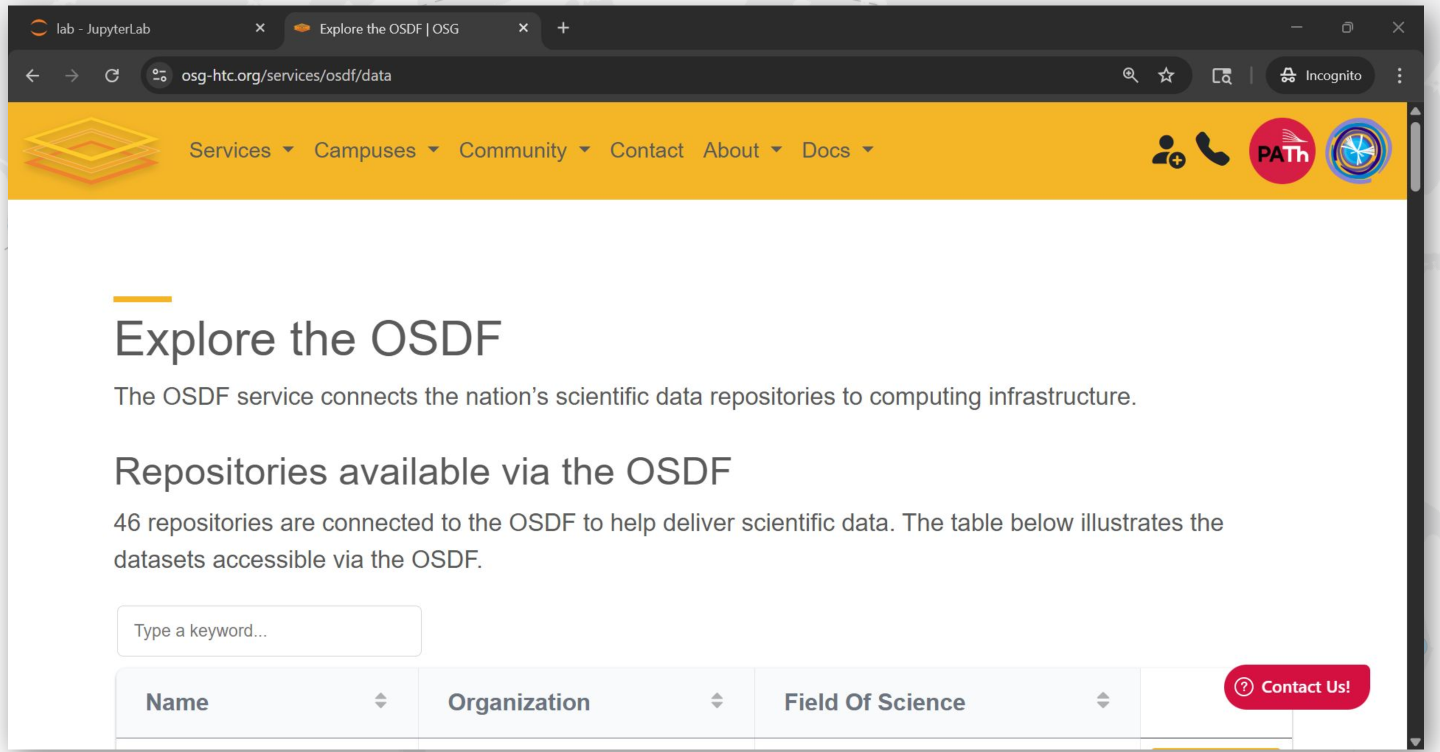
```
(base) jovyan:~$
```

Quickstart

1. Sign in to a Guest notebook at notebook.ospool.osg-htc.org using your institution ID.
2. Open a "Terminal" tab in the Jupyter Notebook
3. In a separate window, go to osg-htc.org/services/osdf/data and choose a repository from the table
4. Click on the repository row and copy the command under "Download a Public Object"
5. Paste and the enter command into "Terminal" tab of Jupyter Notebook

Quickstart

osg-htc.org/services/osdf/data



Explore the OSDF

The OSDF service connects the nation's scientific data repositories to computing infrastructure.

Repositories available via the OSDF

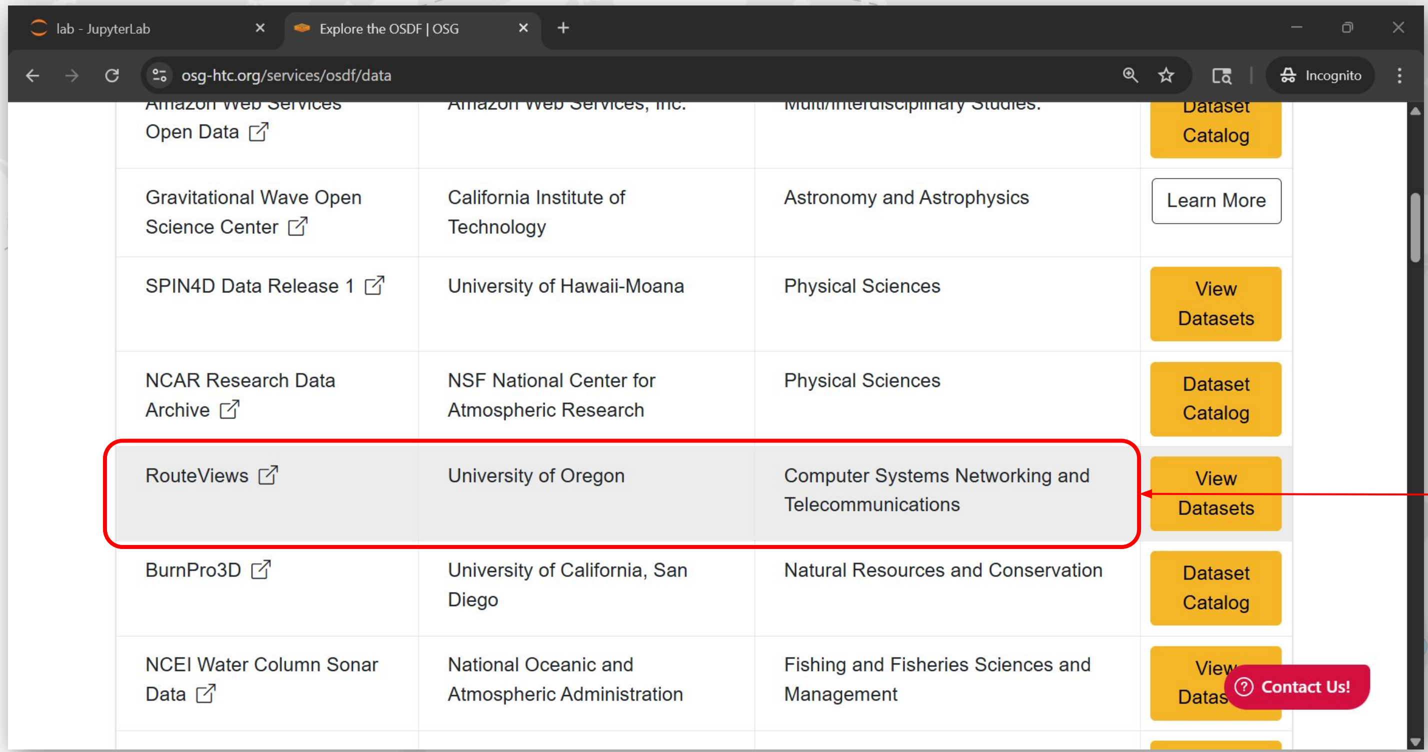
46 repositories are connected to the OSDF to help deliver scientific data. The table below illustrates the datasets accessible via the OSDF.

Type a keyword...

Name	Organization	Field Of Science

Quickstart

osg-htc.org/services/osdf/data

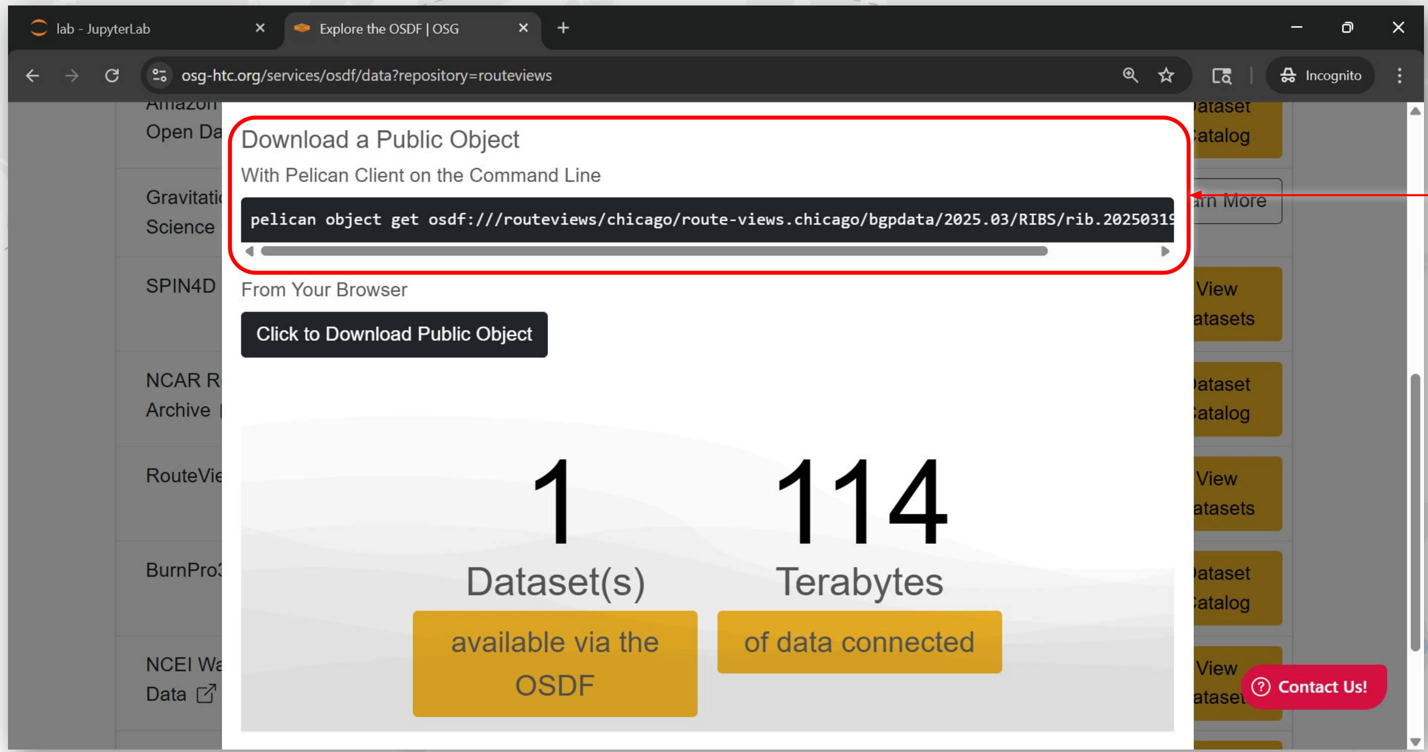


Amazon Web Services Open Data	Amazon Web Services, Inc.	Multi/Interdisciplinary Studies.	Dataset Catalog
Gravitational Wave Open Science Center	California Institute of Technology	Astronomy and Astrophysics	Learn More
SPIN4D Data Release 1	University of Hawaii-Moana	Physical Sciences	View Datasets
NCAR Research Data Archive	NSF National Center for Atmospheric Research	Physical Sciences	Dataset Catalog
RouteViews	University of Oregon	Computer Systems Networking and Telecommunications	View Datasets
BurnPro3D	University of California, San Diego	Natural Resources and Conservation	Dataset Catalog
NCEI Water Column Sonar Data	National Oceanic and Atmospheric Administration	Fishing and Fisheries Sciences and Management	View Datas

Click on a row

Quickstart

osg-htc.org/services/osdf/data



Download a Public Object

With Pelican Client on the Command Line

```
pelican object get osdf:///routeviews/chicago/route-views.chicago/bgpdata/2025.03/RIBS/rib.2025031
```

From Your Browser

Click to Download Public Object

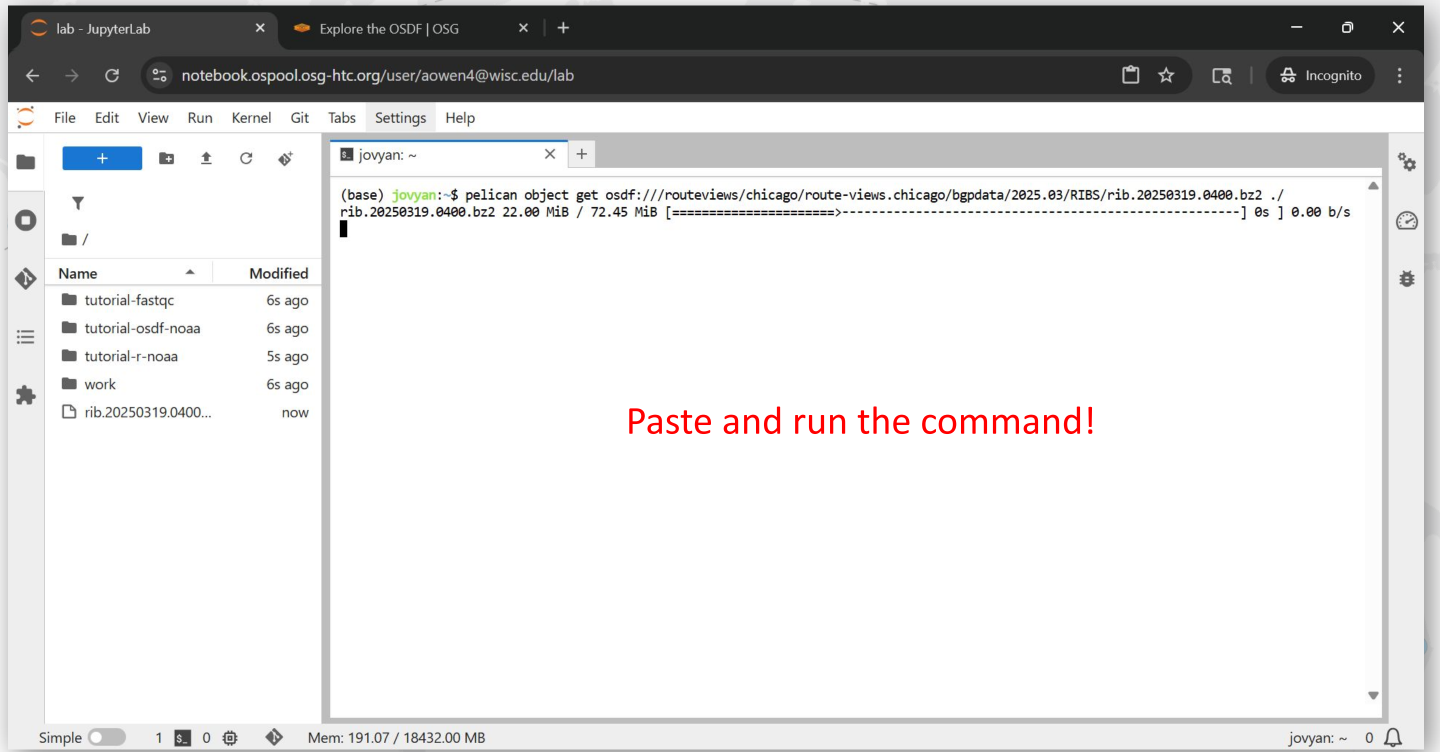
1 Dataset(s) available via the OSDF

114 Terabytes of data connected

Copy the command

Quickstart

notebook.ospool.osg-htc.org



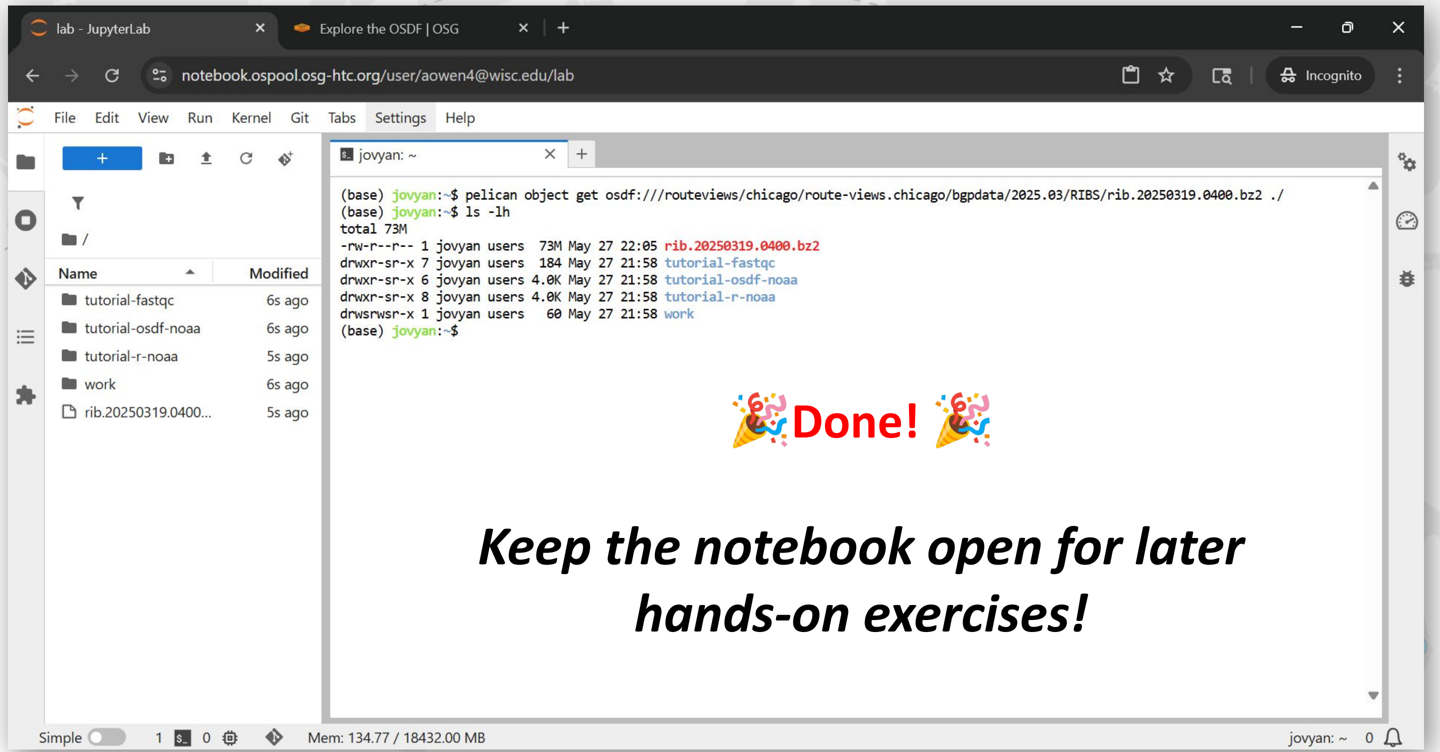
Name	Modified
tutorial-fastqc	6s ago
tutorial-osdf-noaa	6s ago
tutorial-r-noaa	5s ago
work	6s ago
rib.20250319.0400...	now

```
(base) jovyan:~$ pelican object get osdf:///routeviews/chicago/route-views.chicago/bgpdata/2025.03/RIBS/rib.20250319.0400.bz2 ./
rib.20250319.0400.bz2 22.00 MiB / 72.45 MiB [======================>-------------------------------------------------------] 0s ] 0.00 b/s
```

Paste and run the command!

Quickstart

notebook.ospool.osg-htc.org

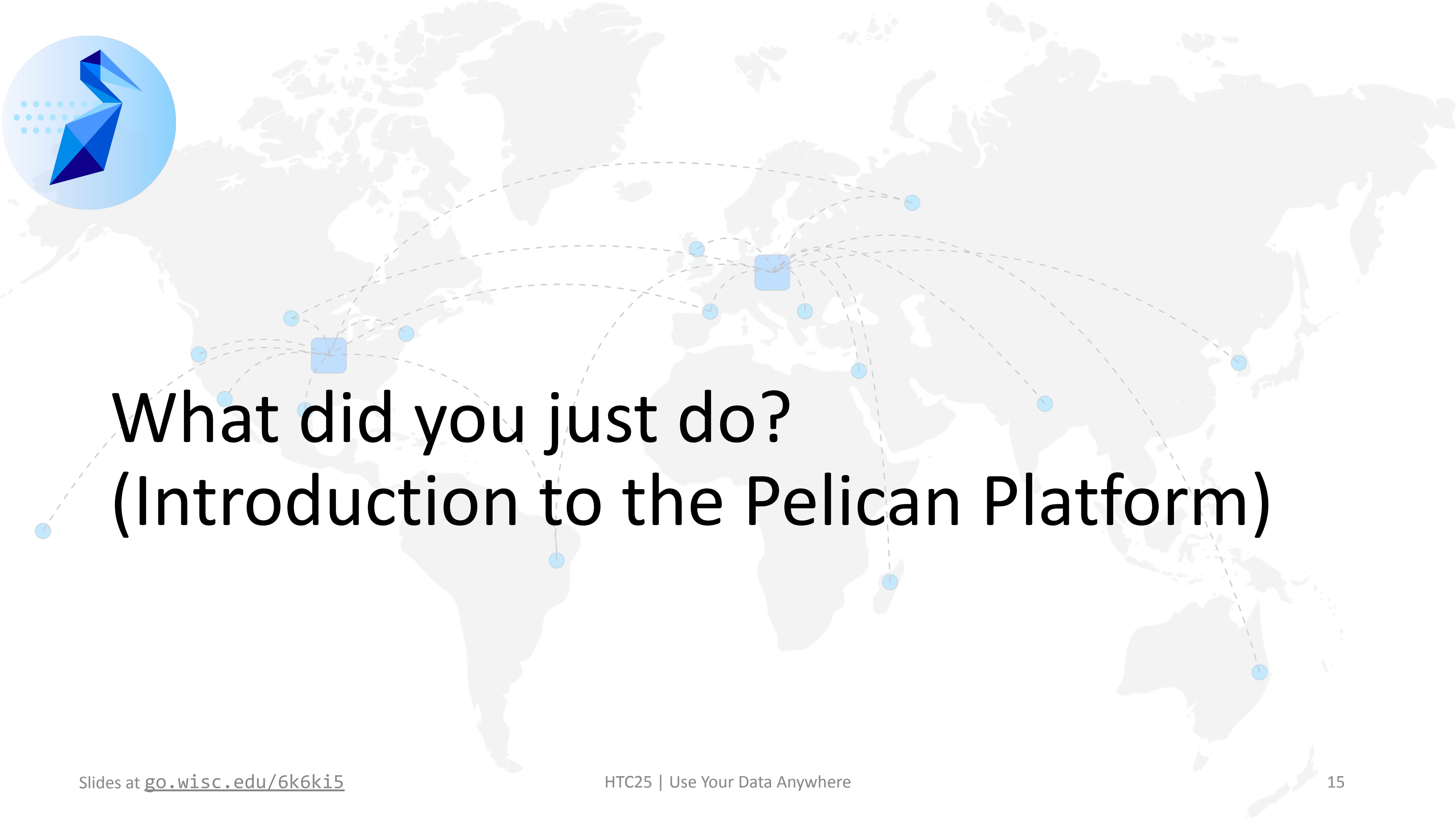

lab - JupyterLab
Explore the OSDF | OSG
notebook.ospool.osg-htc.org/user/aowen4@wisc.edu/lab
Incognito
File Edit View Run Kernel Git Tabs Settings Help
jovyan: ~
Simple
Mem: 134.77 / 18432.00 MB
jovyan: ~

Name	Modified
tutorial-fastqc	6s ago
tutorial-osdf-noaa	6s ago
tutorial-r-noaa	5s ago
work	6s ago
rib.20250319.0400...	5s ago

```
(base) jovyan:~$ pelican object get osdf:///routeviews/chicago/route-views.chicago/bgpdata/2025.03/RIBS/rib.20250319.0400.bz2 ./
(base) jovyan:~$ ls -lh
total 73M
-rw-r--r-- 1 jovyan users  73M May 27 22:05 rib.20250319.0400.bz2
drwxr-sr-x 7 jovyan users  184 May 27 21:58 tutorial-fastqc
drwxr-sr-x 6 jovyan users 4.0K May 27 21:58 tutorial-osdf-noaa
drwxr-sr-x 8 jovyan users 4.0K May 27 21:58 tutorial-r-noaa
drwsrwsr-x 1 jovyan users   60 May 27 21:58 work
(base) jovyan:~$
```

🎉**Done!** 🎉

Keep the notebook open for later hands-on exercises!



What did you just do? (Introduction to the Pelican Platform)

What did you do?

There are three main things that you did:

1. Found a data repository available via the OSDF
2. Found the Pelican URL for that data repository
3. Downloaded the corresponding object using Pelican

Let's examine these steps to get an introduction to Pelican

Exploring the OSDF

The Open Science Data Federation (OSDF) connects disparate dataset repositories into a **single, nationwide data distribution network**.

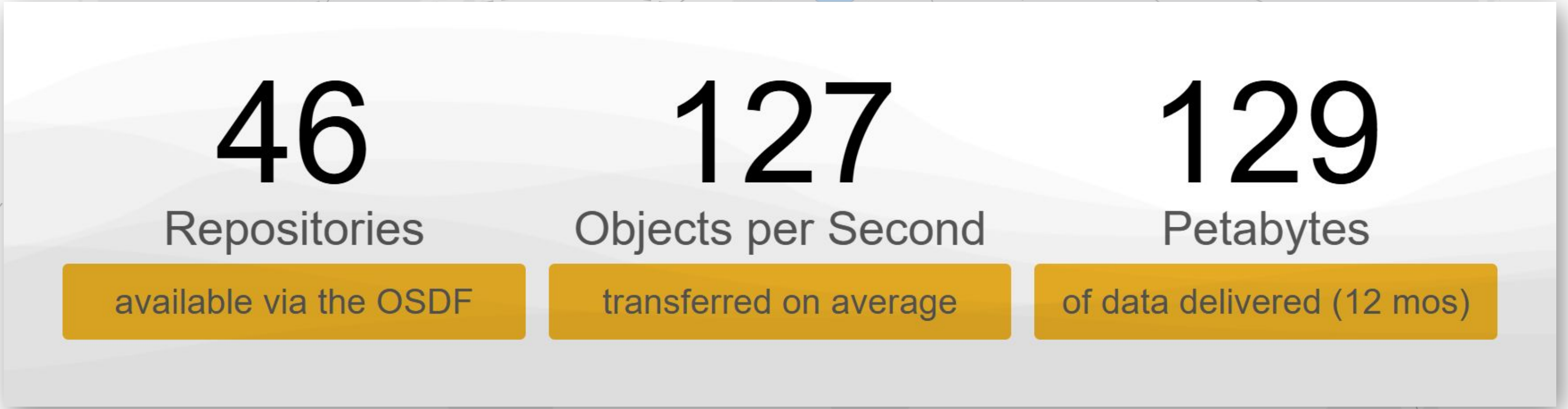
Leveraging the OSDF, **providers can make their datasets available to a wide variety of compute users**, from browsers to Jupyter notebooks to high throughput computing environments.

The OSDF is part of the OSG Fabric of Services, **running software developed by the Pelican Platform**.

https://osg-htc.org/services/osdf

Exploring the OSDF

osg-htc.org/services/osdf/data



46
Repositories
available via the OSDF

127
Objects per Second
transferred on average

129
Petabytes
of data delivered (12 mos)

… and growing!

Exploring the OSDF

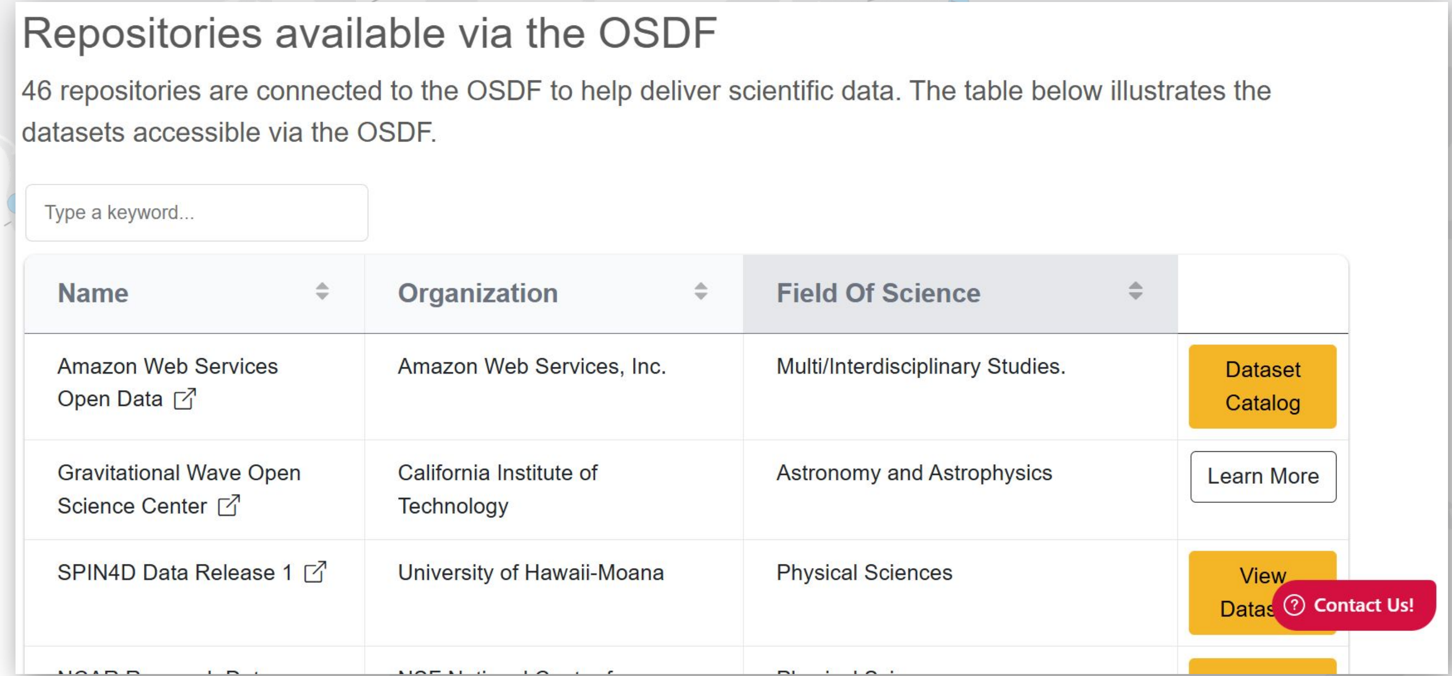
The OSDF is powered by the **Pelican Platform**, a software suite for creating **data federations**, which serves to unite **data contributors** with **data consumers** under a single namespace and access model.

- **Data contributors** can connect existing datastores to a **data federation** while maintaining their access policies

- **Data consumers** can easily access data objects in the **data federation** without needing to know anything about the underlying infrastructure

Exploring the OSDF

osg-htc.org/services/osdf/data



Repositories available via the OSDF

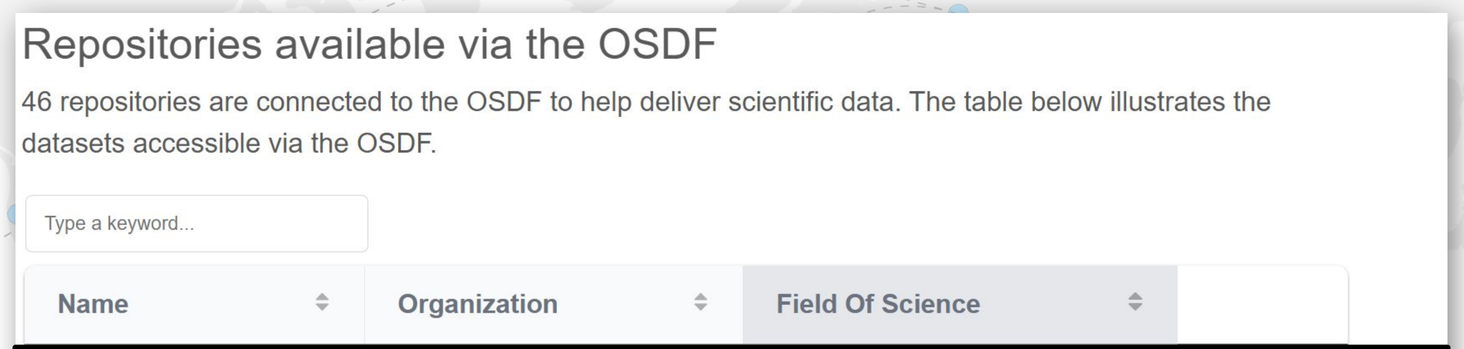
46 repositories are connected to the OSDF to help deliver scientific data. The table below illustrates the datasets accessible via the OSDF.

Type a keyword...

Name	Organization	Field Of Science	
Amazon Web Services Open Data	Amazon Web Services, Inc.	Multi/Interdisciplinary Studies.	Dataset Catalog
Gravitational Wave Open Science Center	California Institute of Technology	Astronomy and Astrophysics	Learn More
SPIN4D Data Release 1	University of Hawaii-Moana	Physical Sciences	View Datas

Exploring the OSDF

osg-htc.org/services/osdf/data


Repositories available via the OSDF

46 repositories are connected to the OSDF to help deliver scientific data. The table below illustrates the datasets accessible via the OSDF.

Type a keyword...

Name	Organization	Field Of Science

Your repository could be in this list!

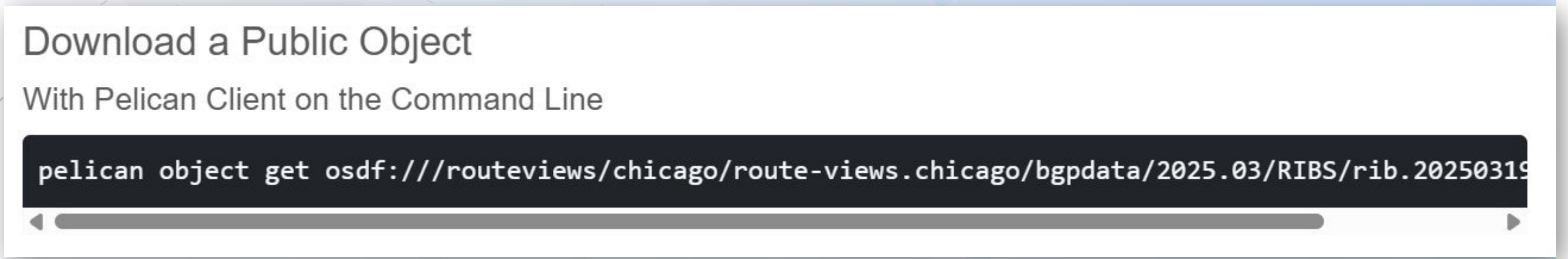
Please reach out if you are even a tiny bit interested: support@osg-htc.org

Exploring the OSDF

osg-htc.org/services/osdf/data

Every repository gets a unique "Pelican URL"

Data consumers only need the Pelican URL to access the data!



Download a Public Object

With Pelican Client on the Command Line

```
pelican object get osdf:///routeviews/chicago/route-views.chicago/bgpdata/2025.03/RIBS/rib.2025031
```

Anatomy of a Pelican URL

The tale of Pelican in 4 parts:

- The "**protocol**"
- The "**discovery URL**"
- The "**namespace prefix**" (aka "**namespace**")
- The "**object name**"

For example:

```
pelican://osg-htc.org/pelicanplatform/test/hello-world.txt
```

Anatomy of a Pelican URL

The "**protocol**"

- Open source web protocol **pelican**
- Extension of HTTP
- Anyone can use it to interact with a Pelican Data Federation!

```
pelican://_______________________________________________
```

Anatomy of a Pelican URL

The "**discovery URL**"

- The **pelican** protocol is used to interact with a Pelican Federation
- The **discovery URL** tells the protocol **which** Federation to talk to!

The **discovery URL** for the OSDF is **osg-htc.org**

```
pelican://osg-htc.org/____________________________________
```

Anatomy of a Pelican URL

The OSDF "protocol"

- Pelican provides a shortcut "protocol" for specifying the OSDF!

`osdf:///_______________________________________`

is exactly equivalent to

`pelican://osg-htc.org/_______________________________________`

Anatomy of a Pelican URL

The "**discovery URL**"

- The **pelican** protocol is used to interact with a Pelican Federation
- The **discovery URL** tells the protocol **which** Federation to talk to!

The **discovery URL** for the OSDF is **osg-htc.org**

```
pelican://osg-htc.org/____________________________________
             osdf:///____________________________________
```

Anatomy of a Pelican URL

The "**namespace prefix**" (aka "**namespace**")

- When a **data provider** connects their repository to a Federation, they make it available under a particular **namespace**
- The **data provider** gets to decide who gets to access their data and has control over all Pelican URLs that start with the same prefix!

The Pelican Platform owns the **pelicanplatform** namespace in the OSDF

```
pelican://osg-htc.org/pelicanplatform/____________________
                osdf:///pelicanplatform/____________________
```

Anatomy of a Pelican URL

The "**object name**"

- Corresponds to data in the repository accessible via the **namespace** connected to the Federation
- The process of connecting the **namespace** to the Federation includes mapping the storage into individual object names

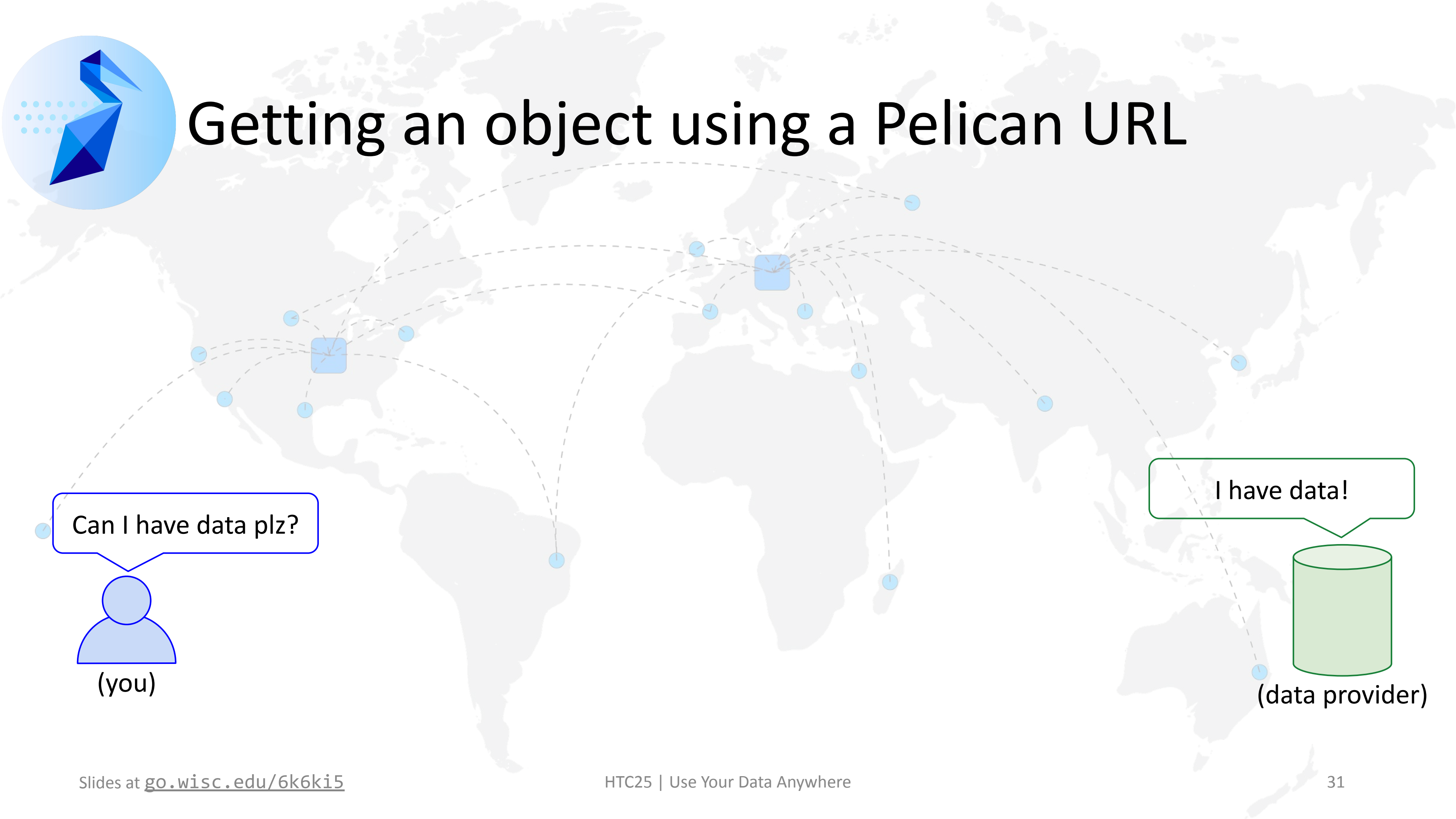
```
pelican://osg-htc.org/pelicanplatform/test/hello-world.txt

                osdf:///pelicanplatform/test/hello-world.txt
```

Getting an object using a Pelican URL

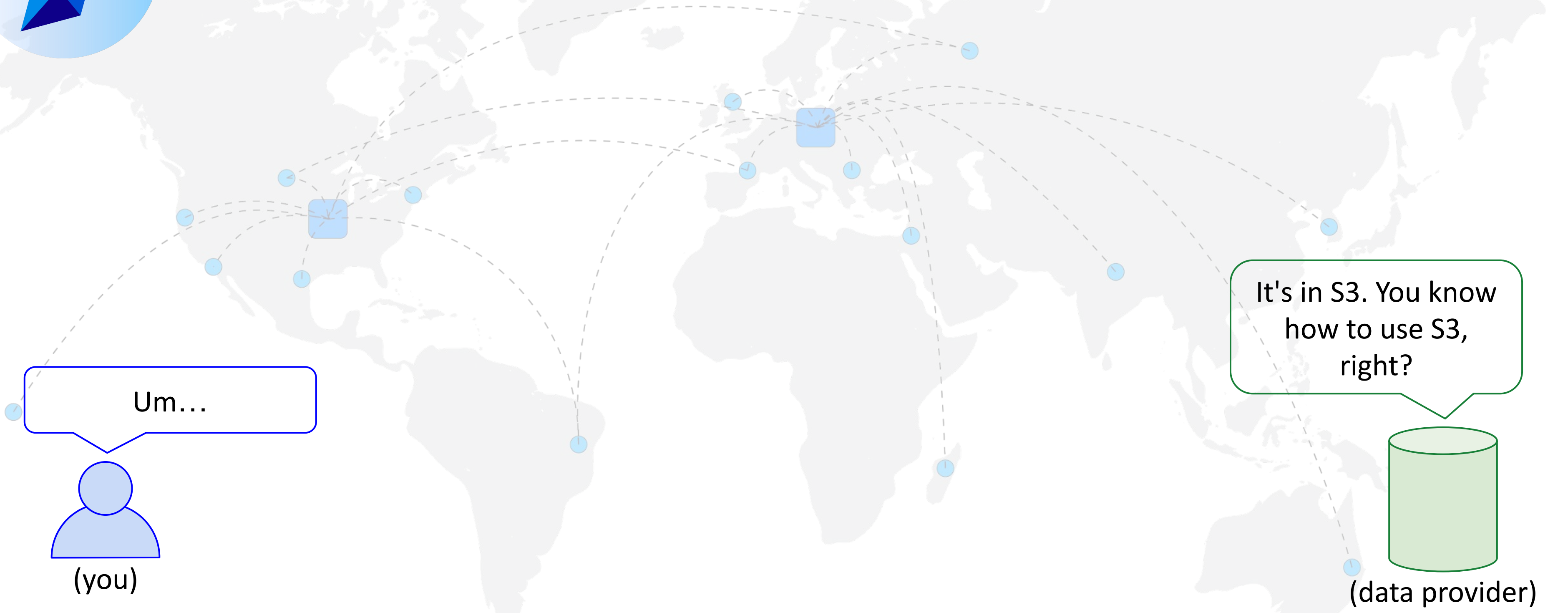
If you know the Pelican URL for an object, you can download (get) it!

But how?

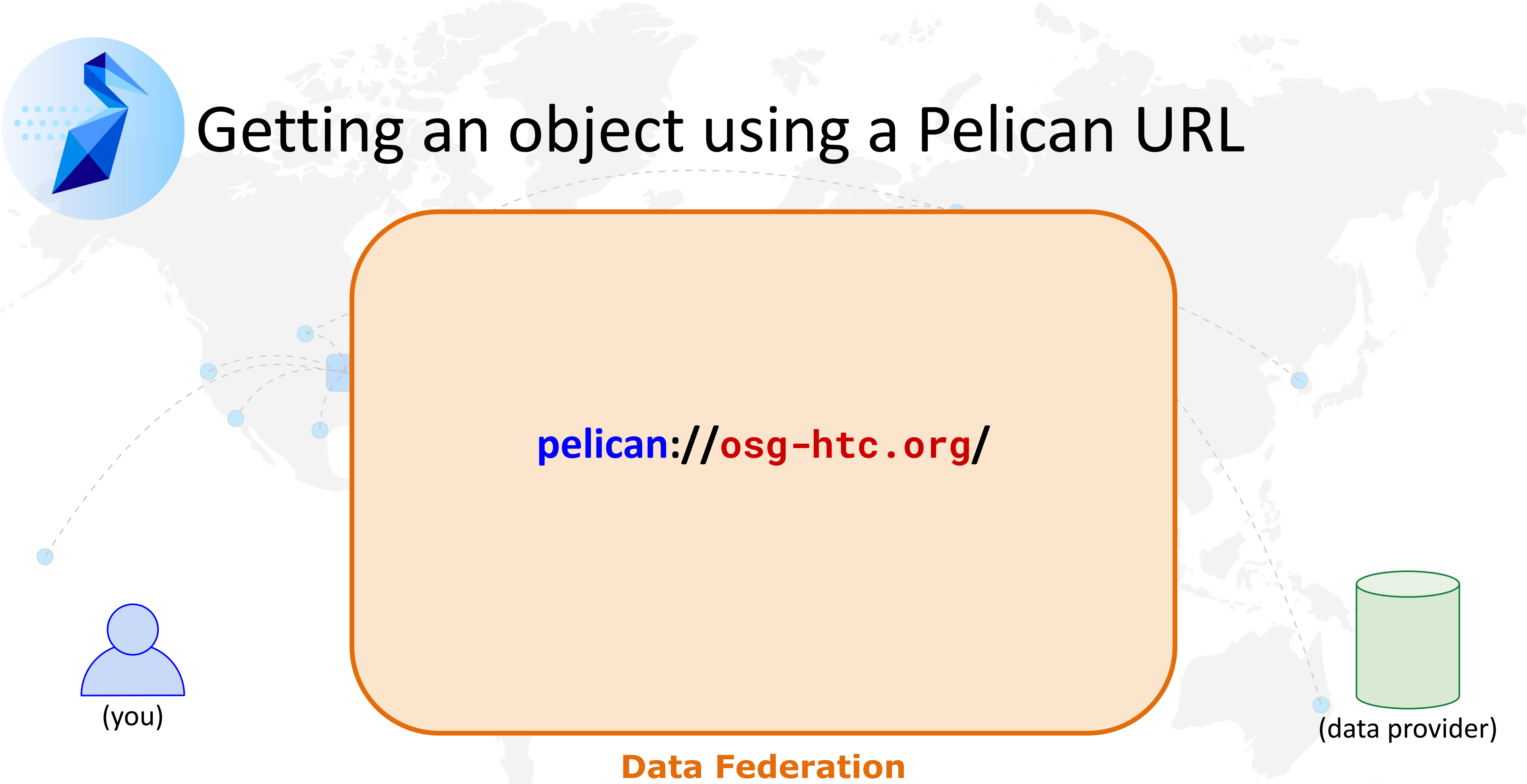
Getting an object using a Pelican URL



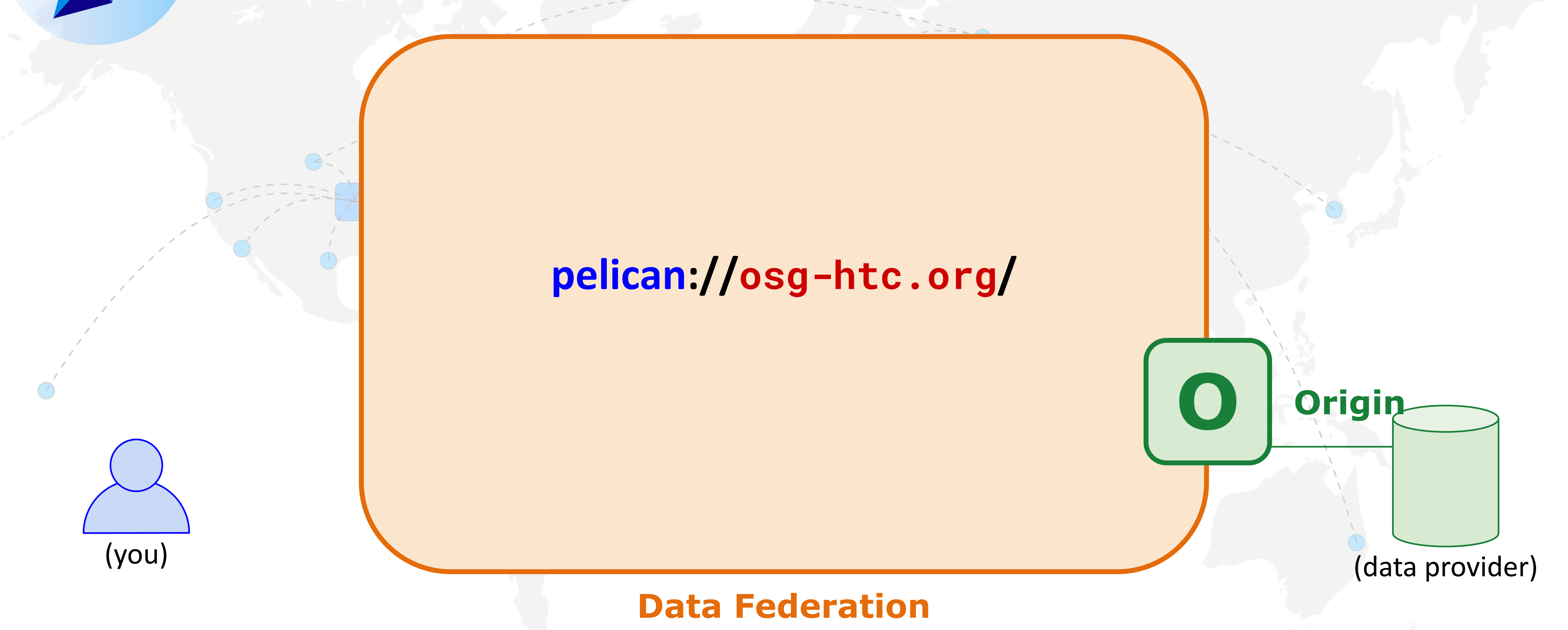
Getting an object using a Pelican URL



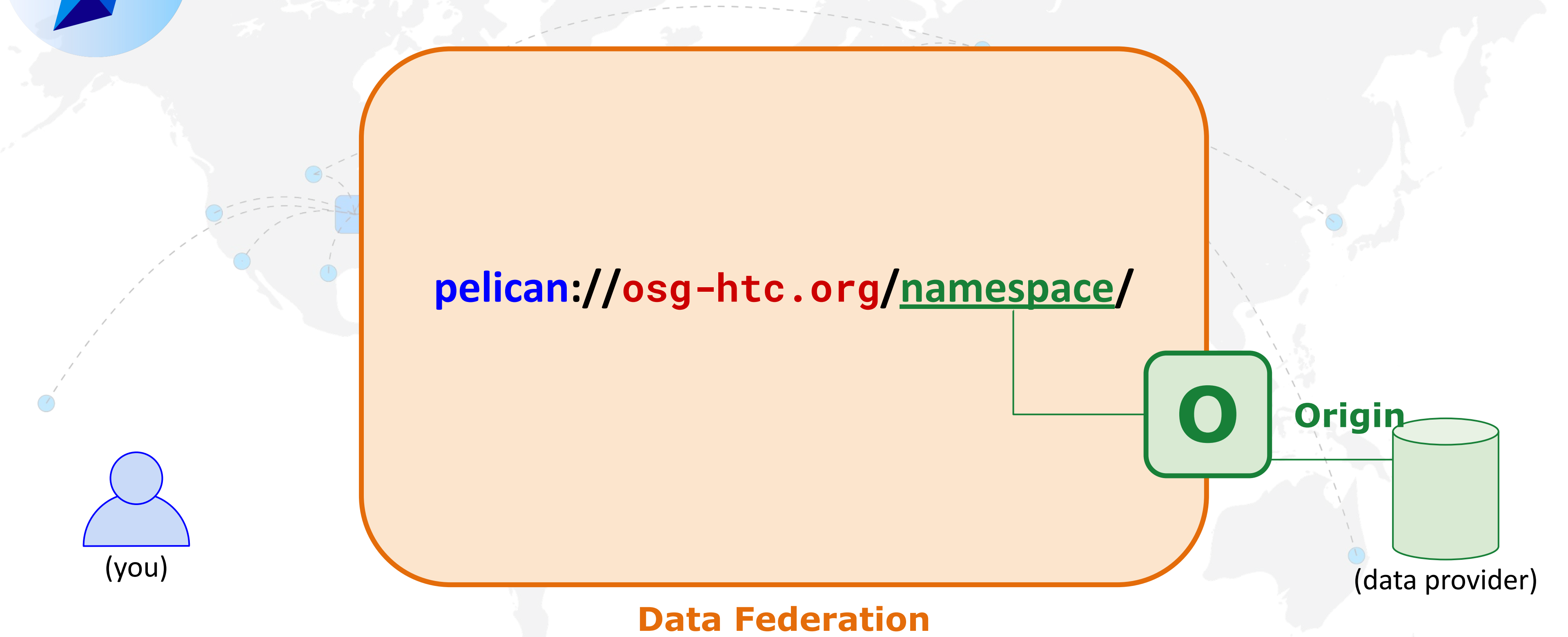
Getting an object using a Pelican URL



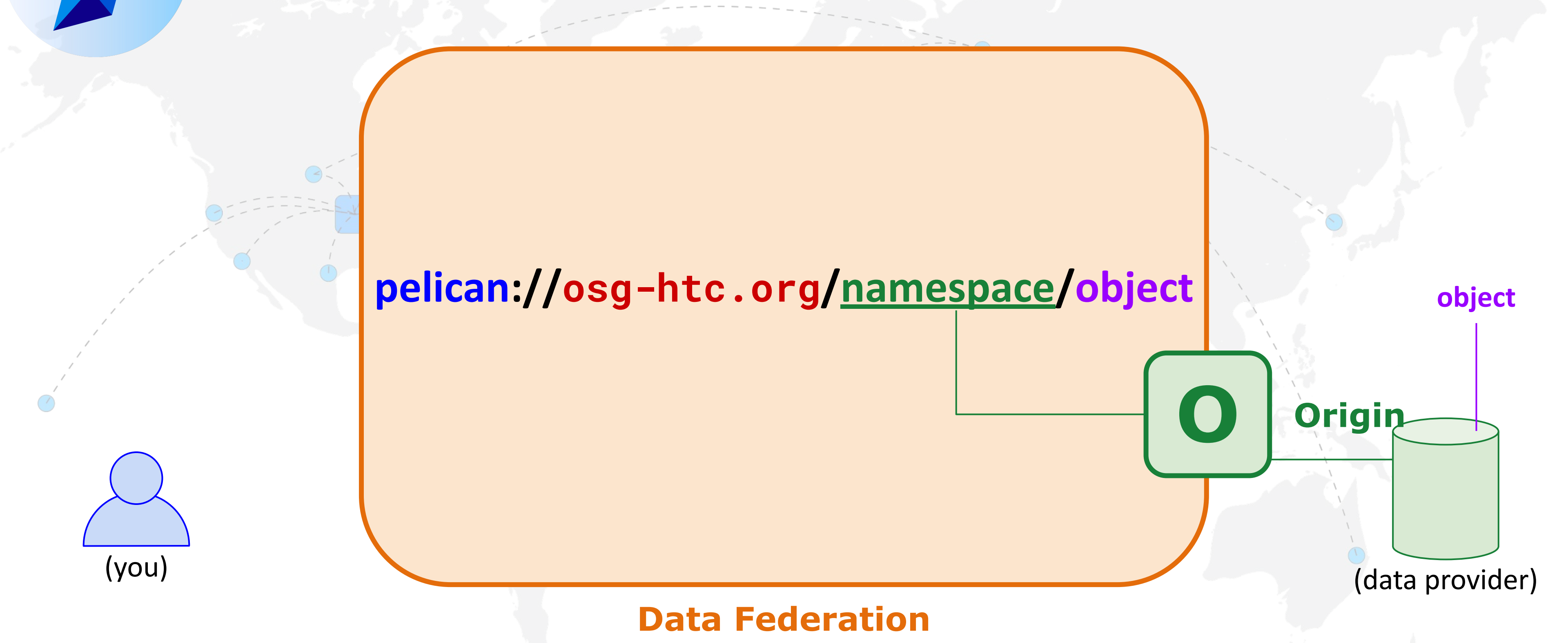
Getting an object using a Pelican URL



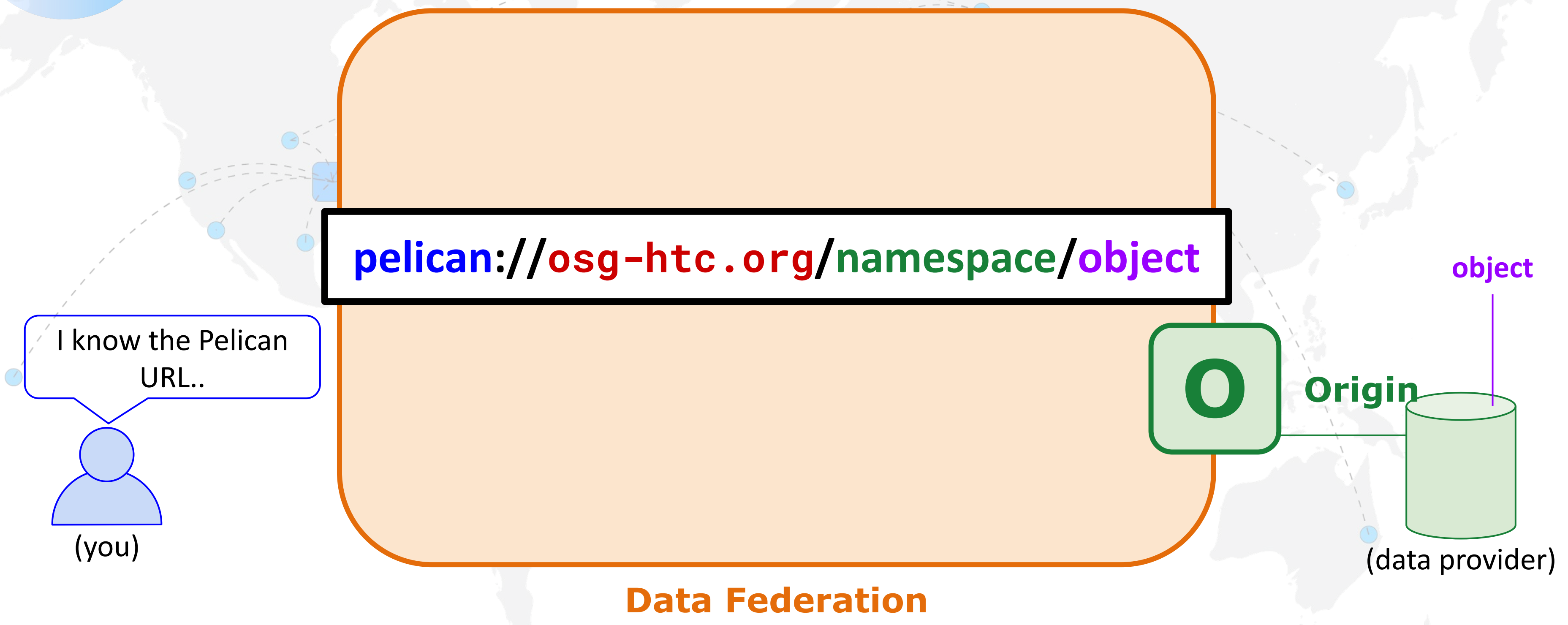
Getting an object using a Pelican URL



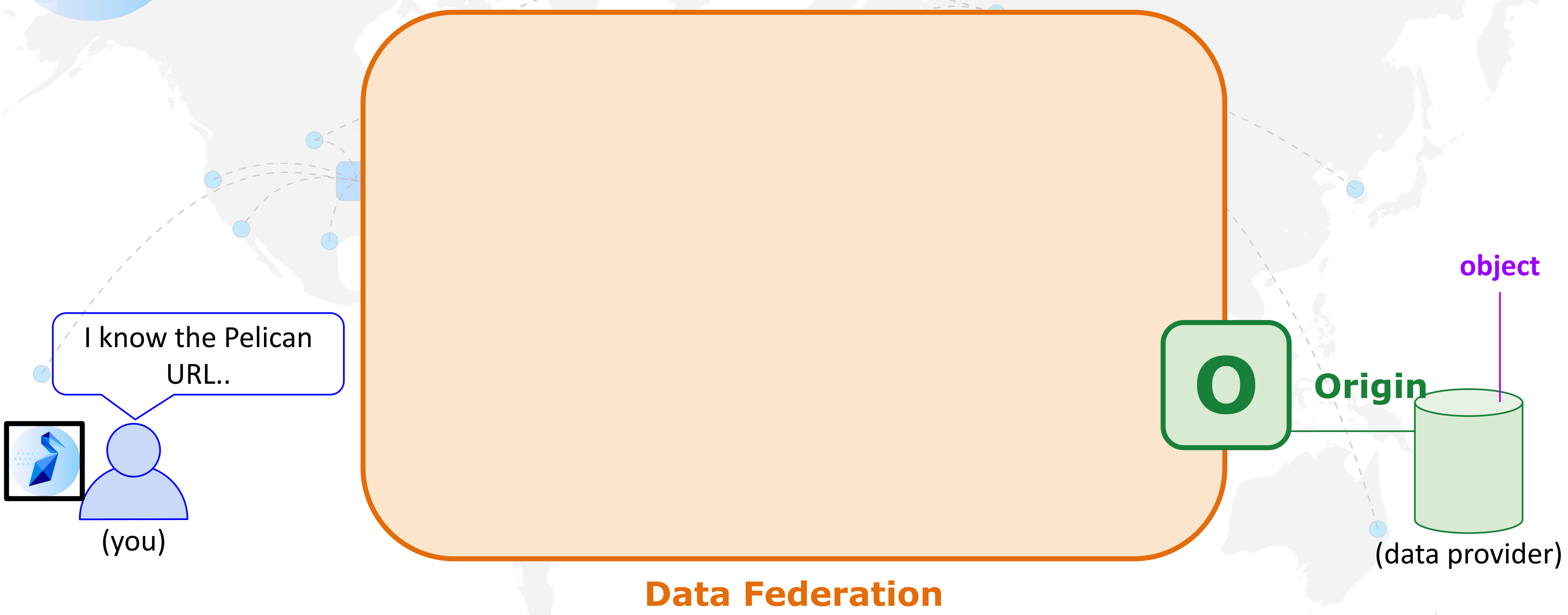
Getting an object using a Pelican URL



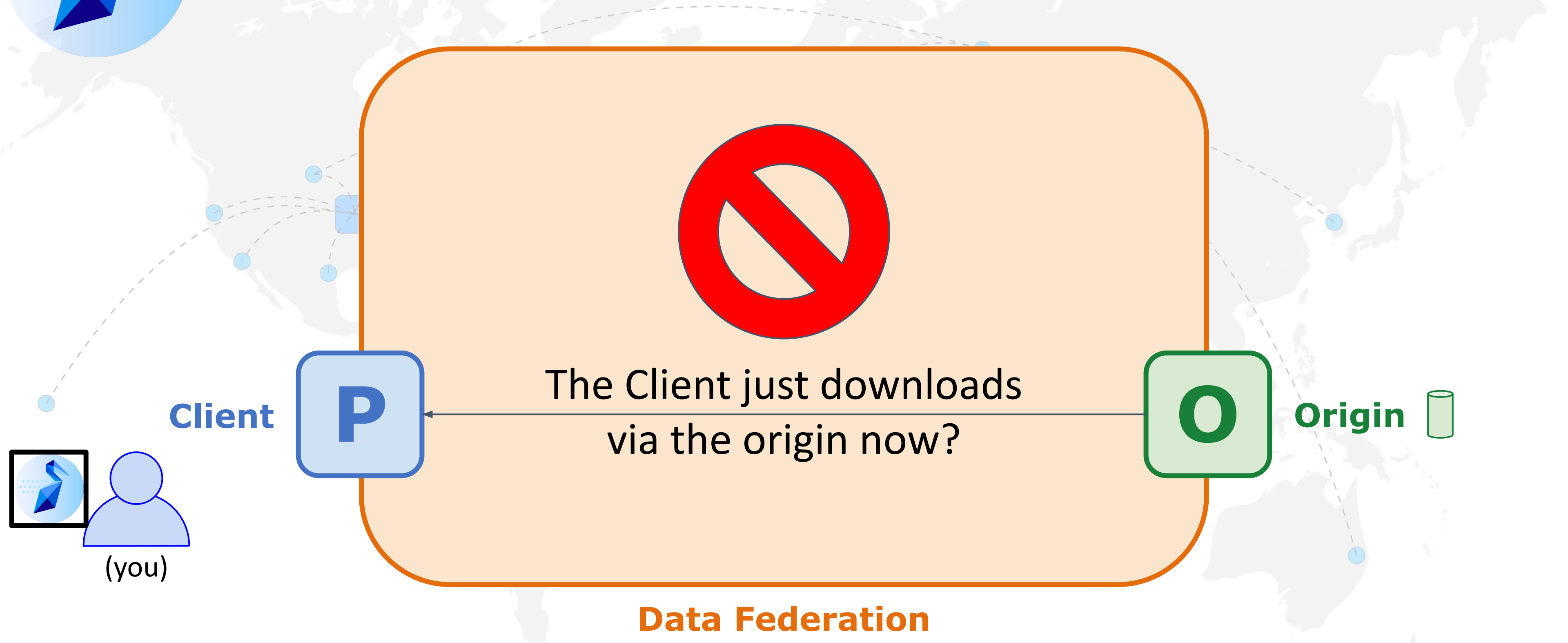
Getting an object using a Pelican URL



Getting an object using a Pelican URL

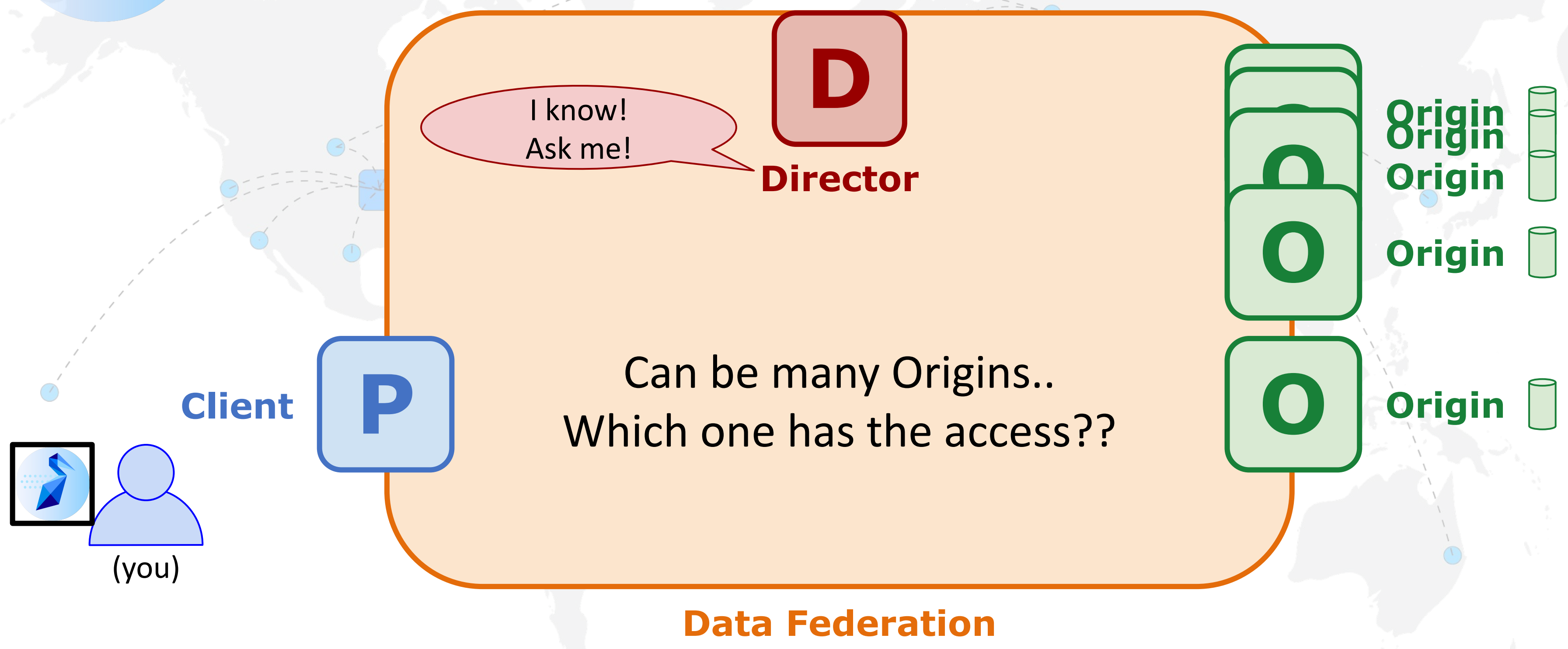


Getting an object using a Pelican URL



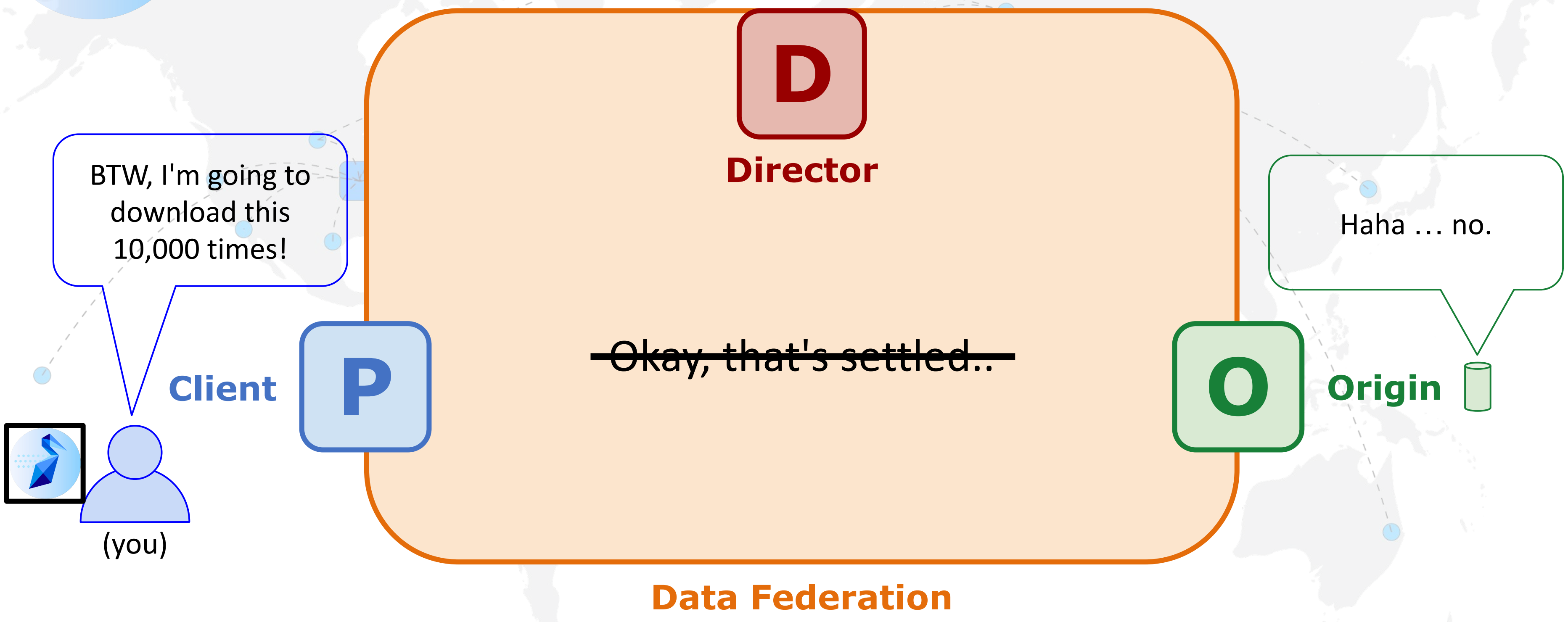
The Client just downloads via the origin now?

Getting an object using a Pelican URL

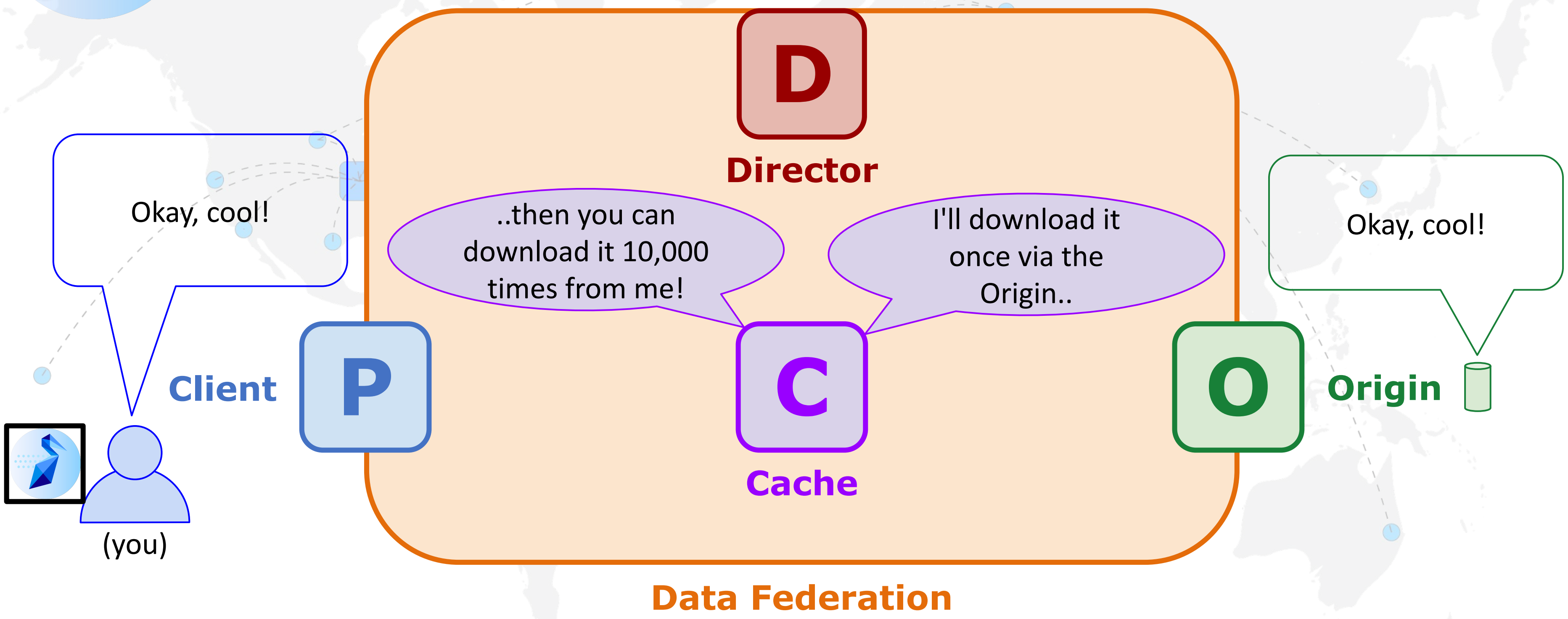


Can be many Origins..
Which one has the access??

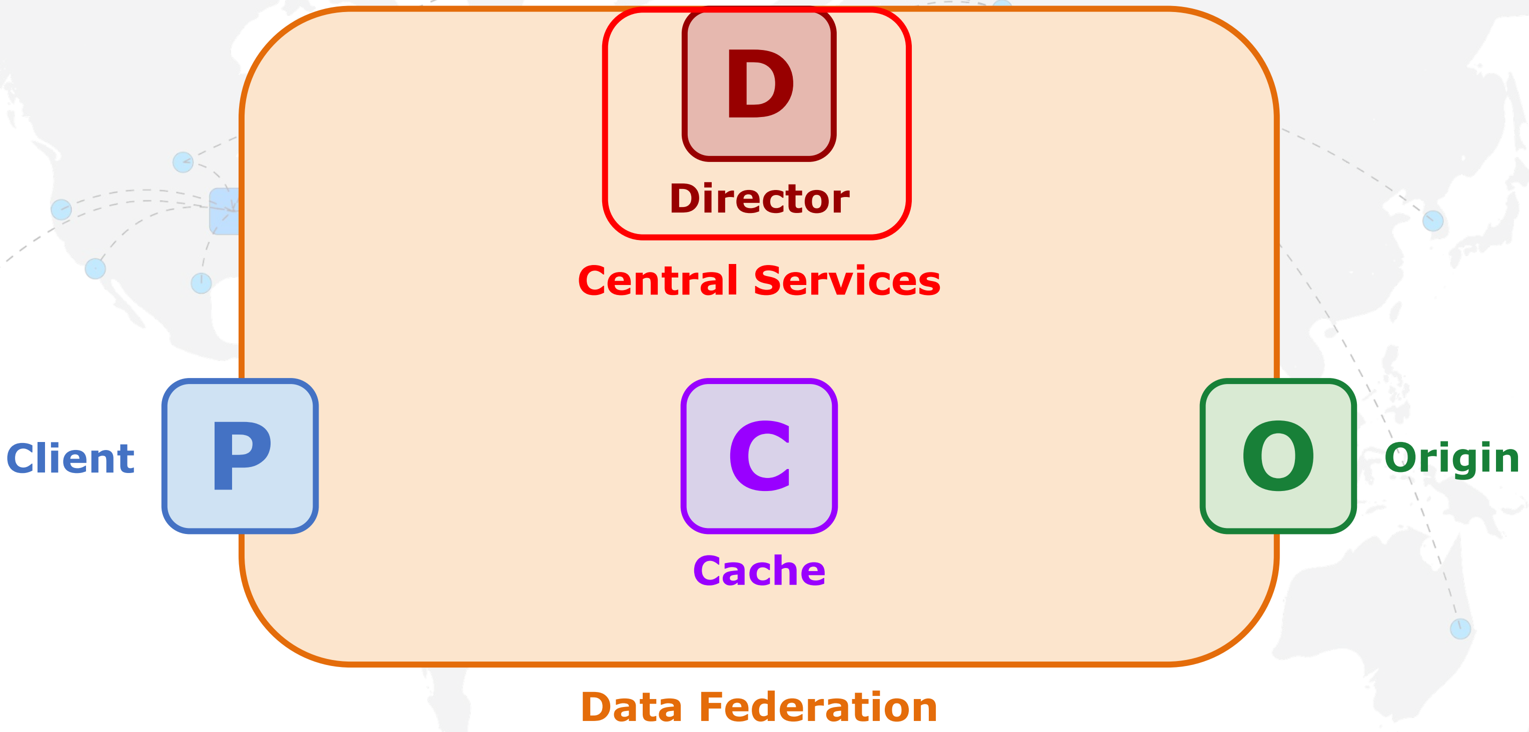
Getting an object using a Pelican URL


BTW, I'm going to download this 10,000 times!
Director
Client
~~Okay, that's settled..~~
Origin
Haha … no.
(you)
Data Federation

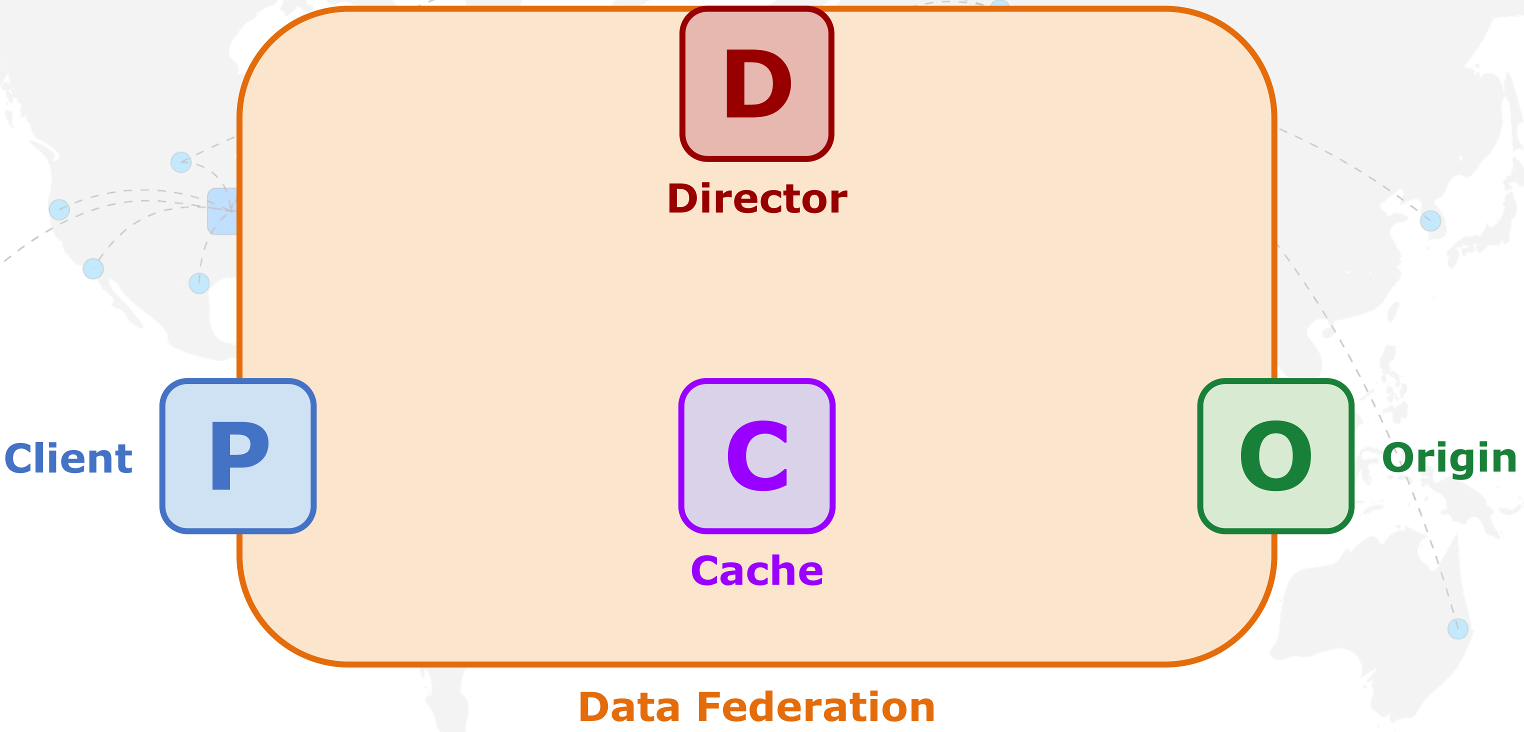
Getting an object using a Pelican URL


Okay, cool!
Client
(you)
P
D
Director
..then you can download it 10,000 times from me!
I'll download it once via the Origin..
C
Cache
O
Origin
Okay, cool!
Data Federation

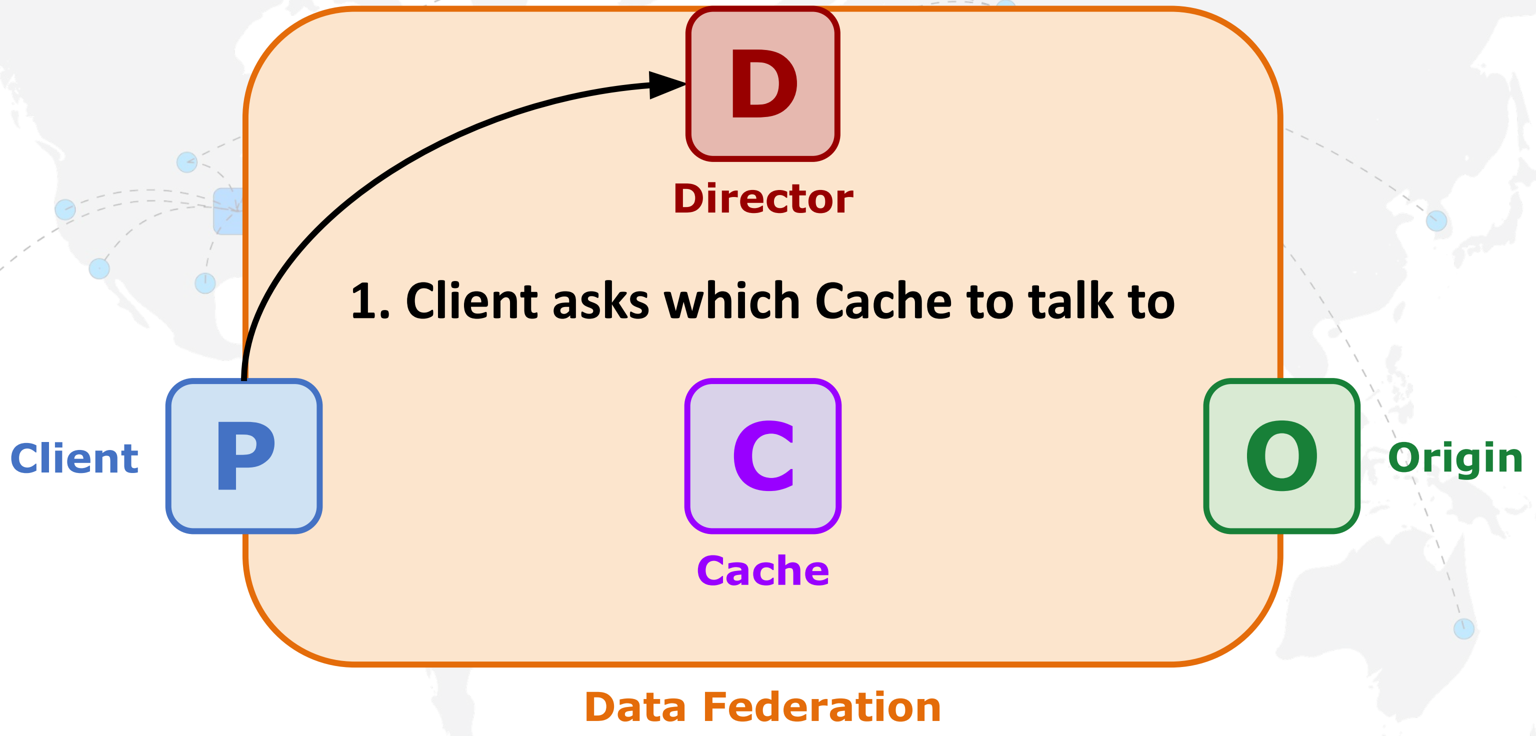
Getting an object using a Pelican URL



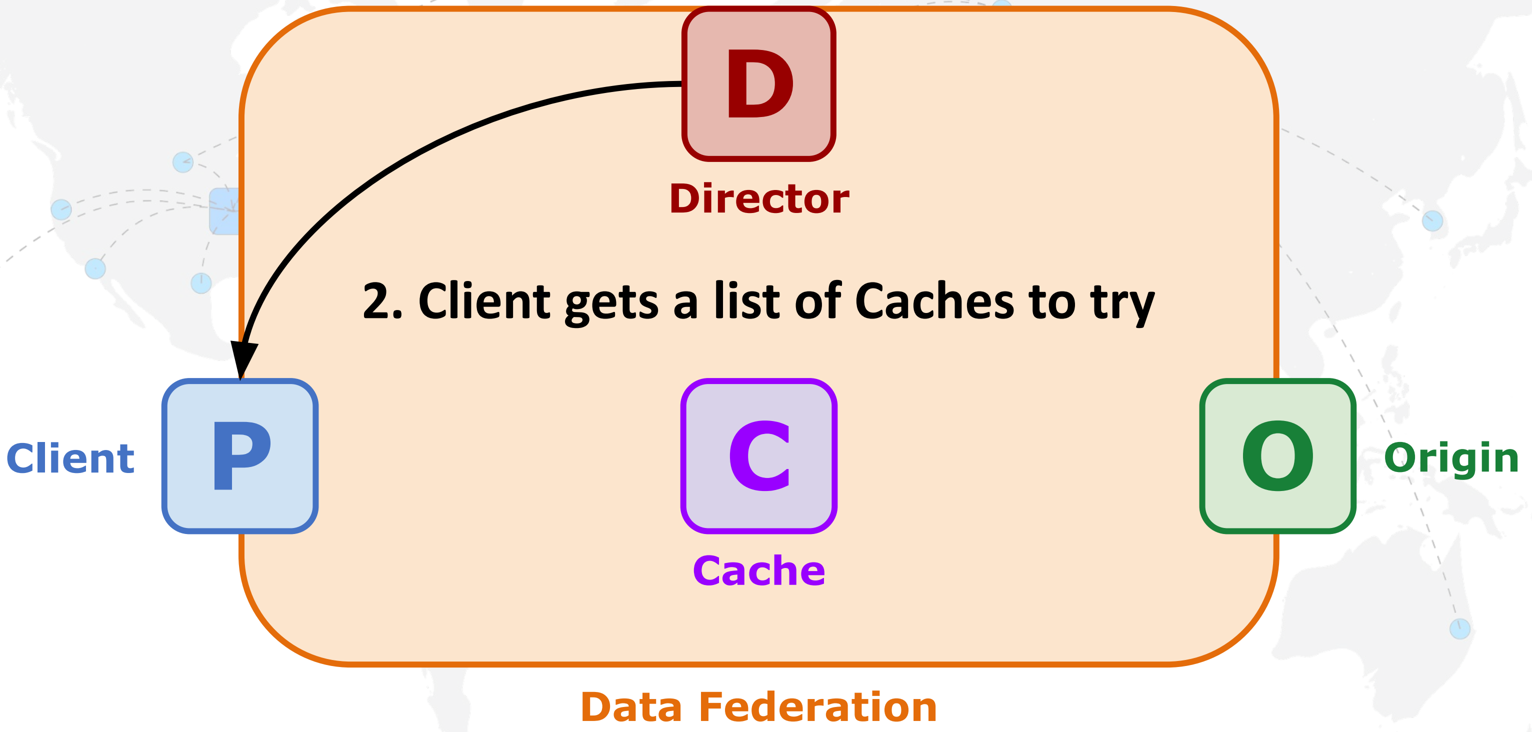
Getting an object using a Pelican URL



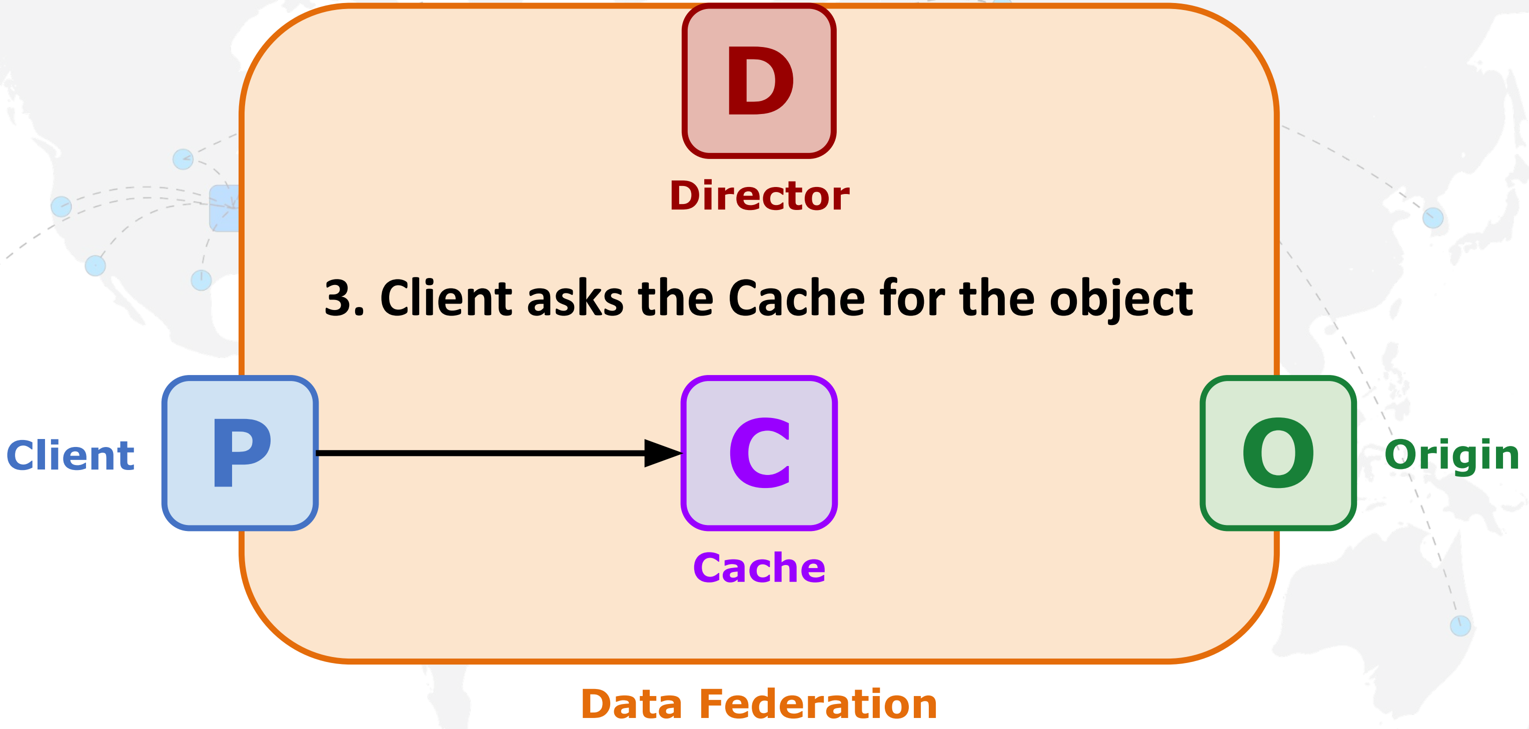
Getting an object using a Pelican URL



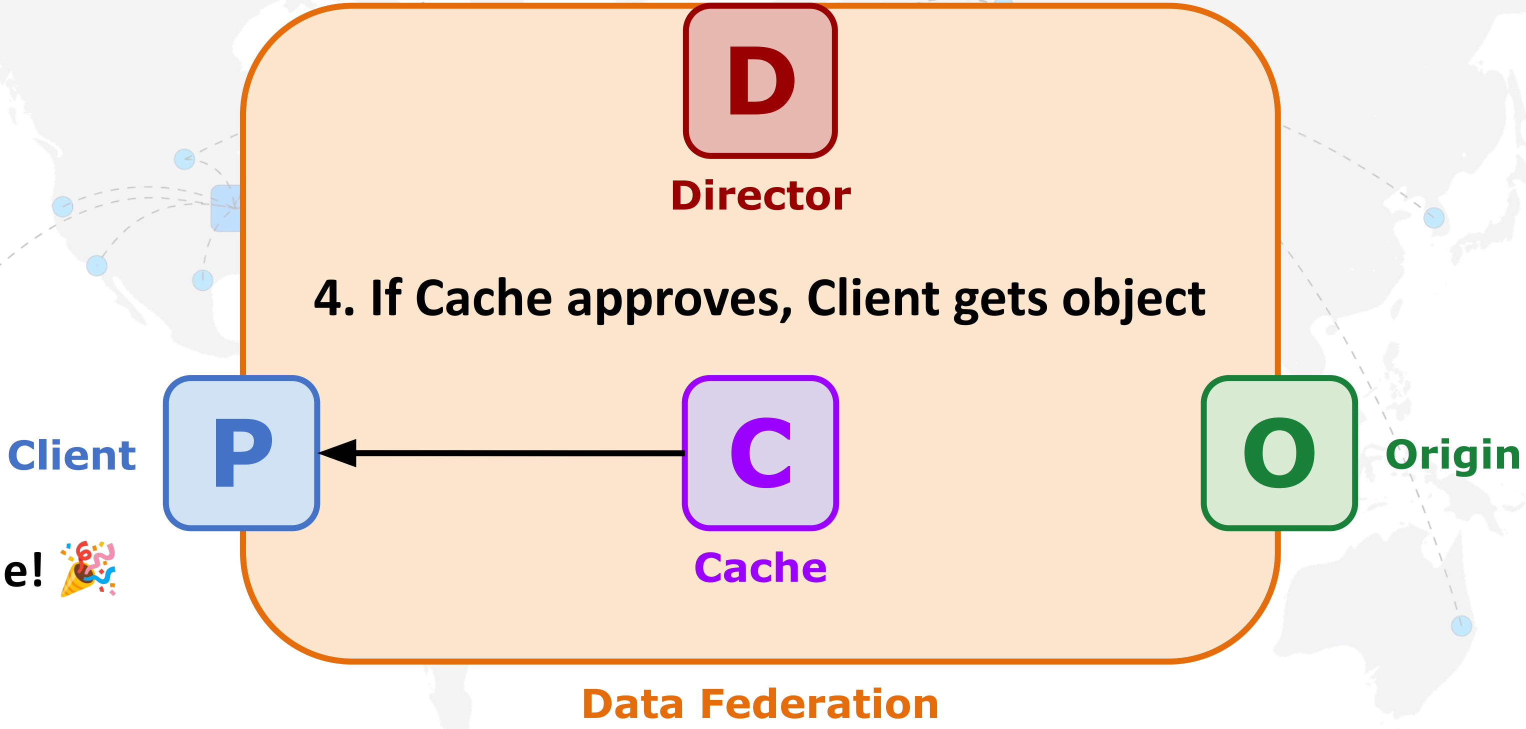
Getting an object using a Pelican URL



Getting an object using a Pelican URL

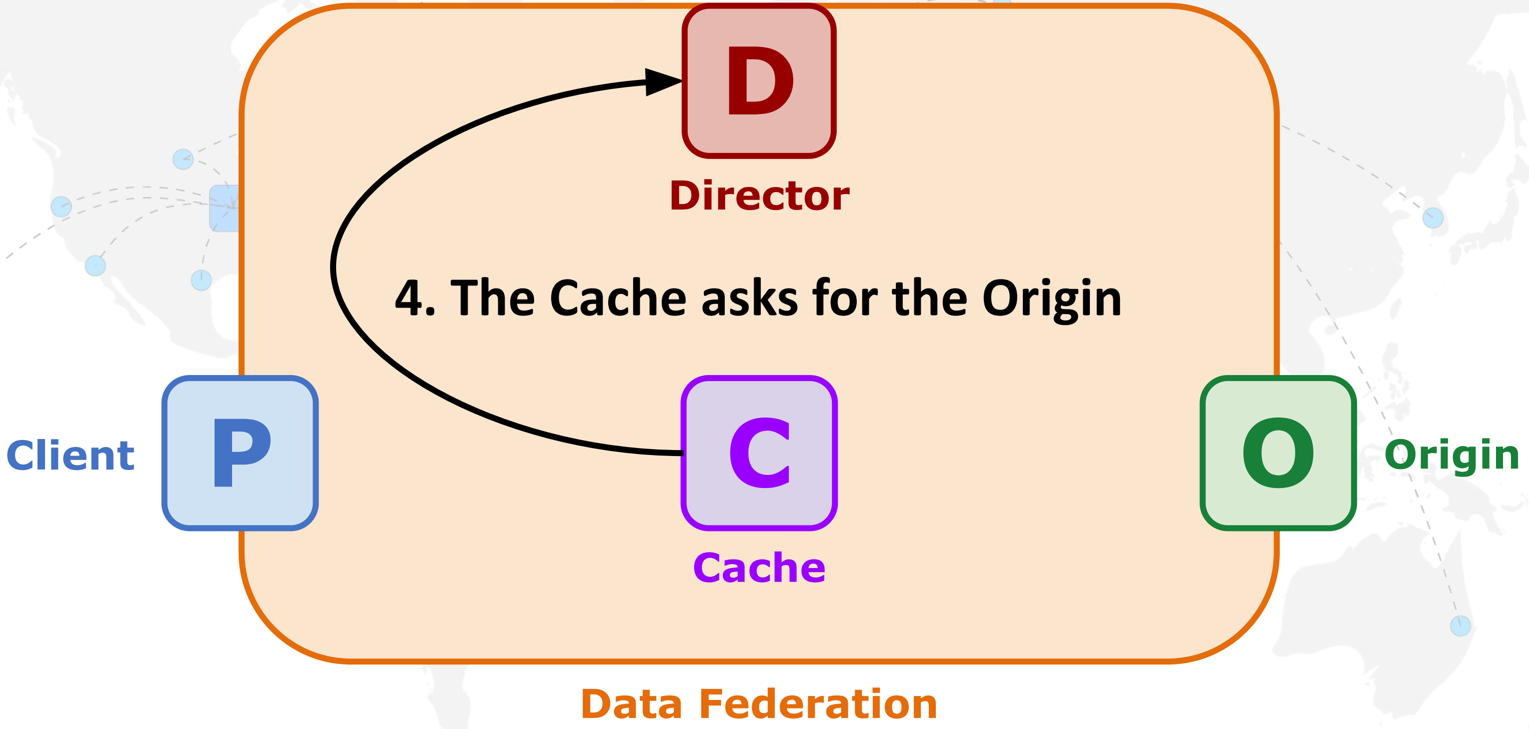


a) The Cache has the object

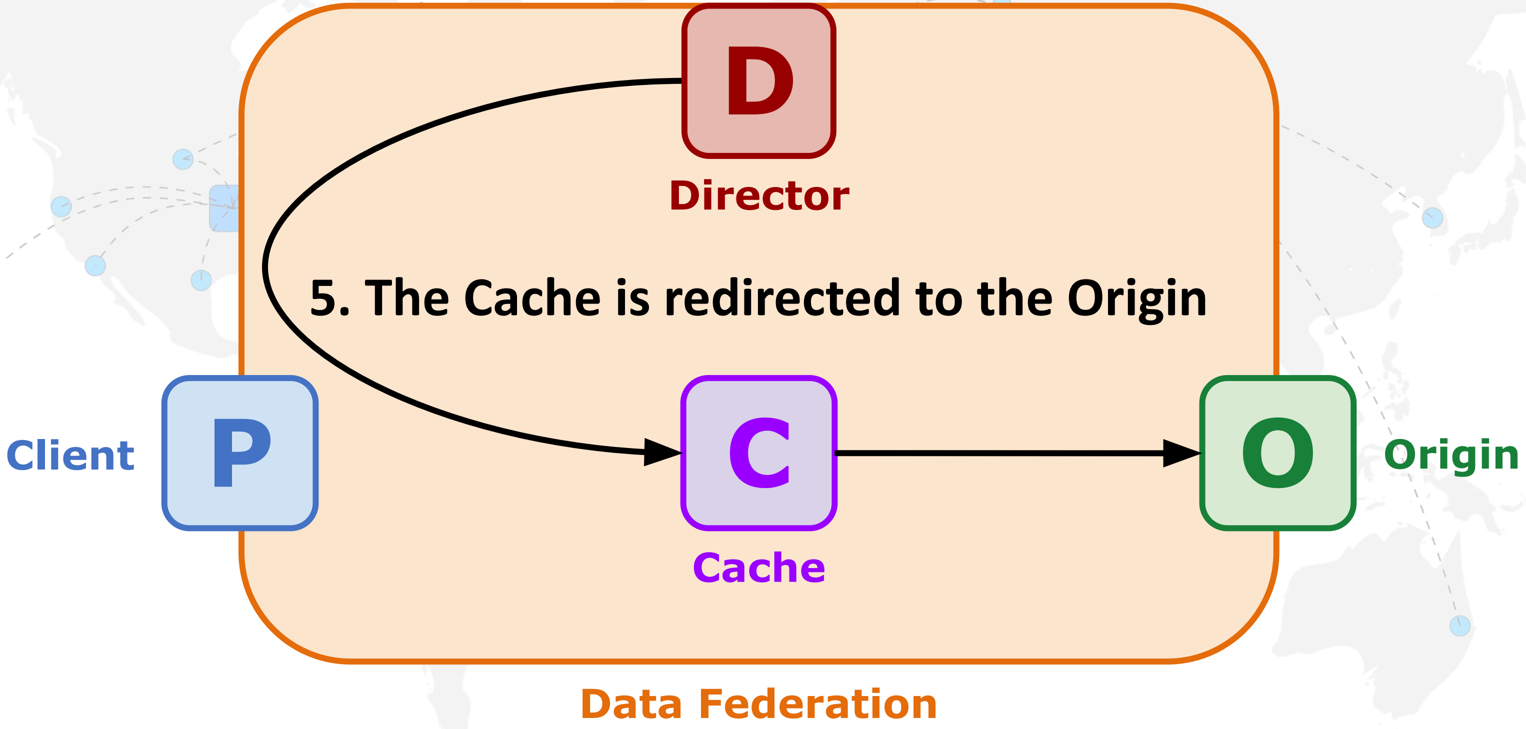


🎉Done! 🎉

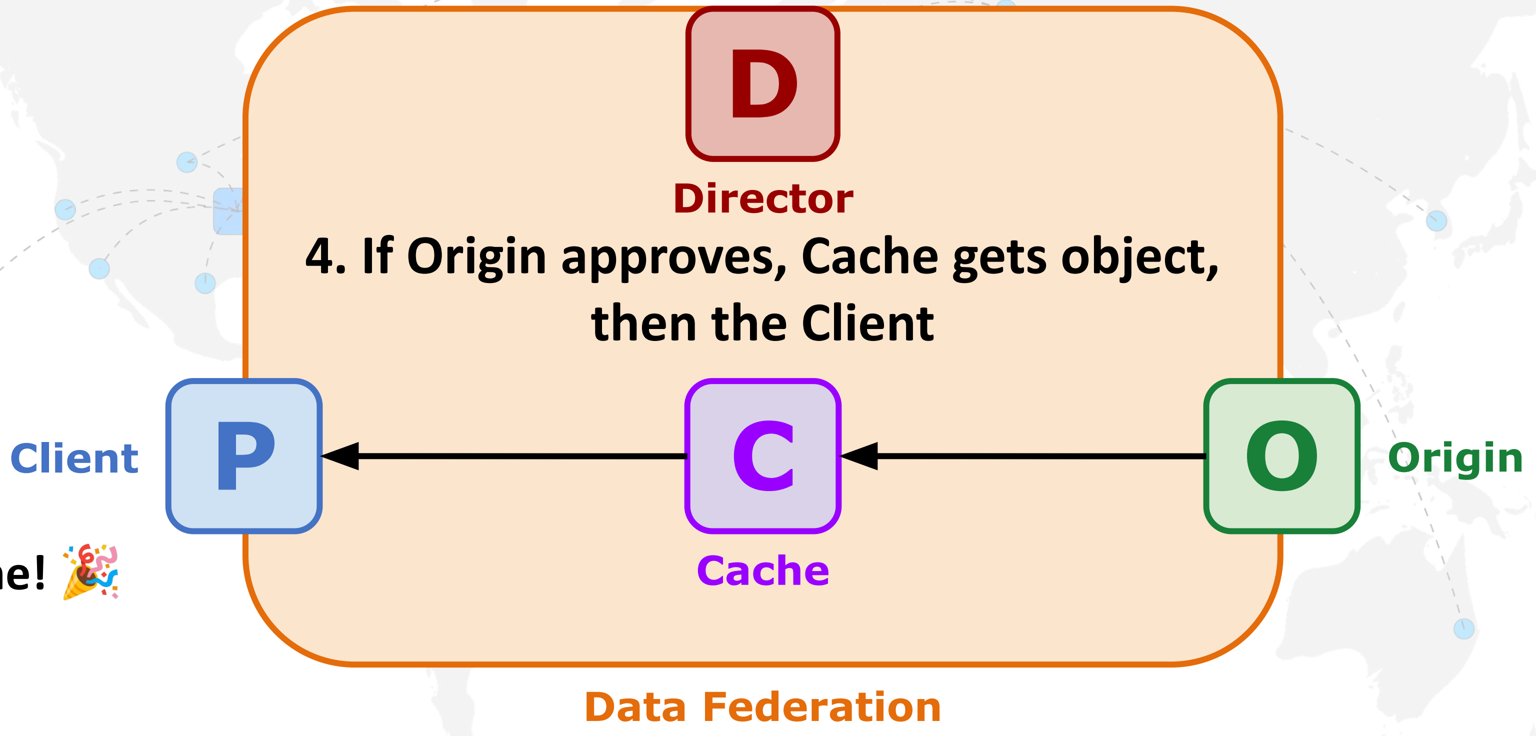
b) The Cache does not have the object



b) The Cache does not have the object



b) The Cache does not have the object

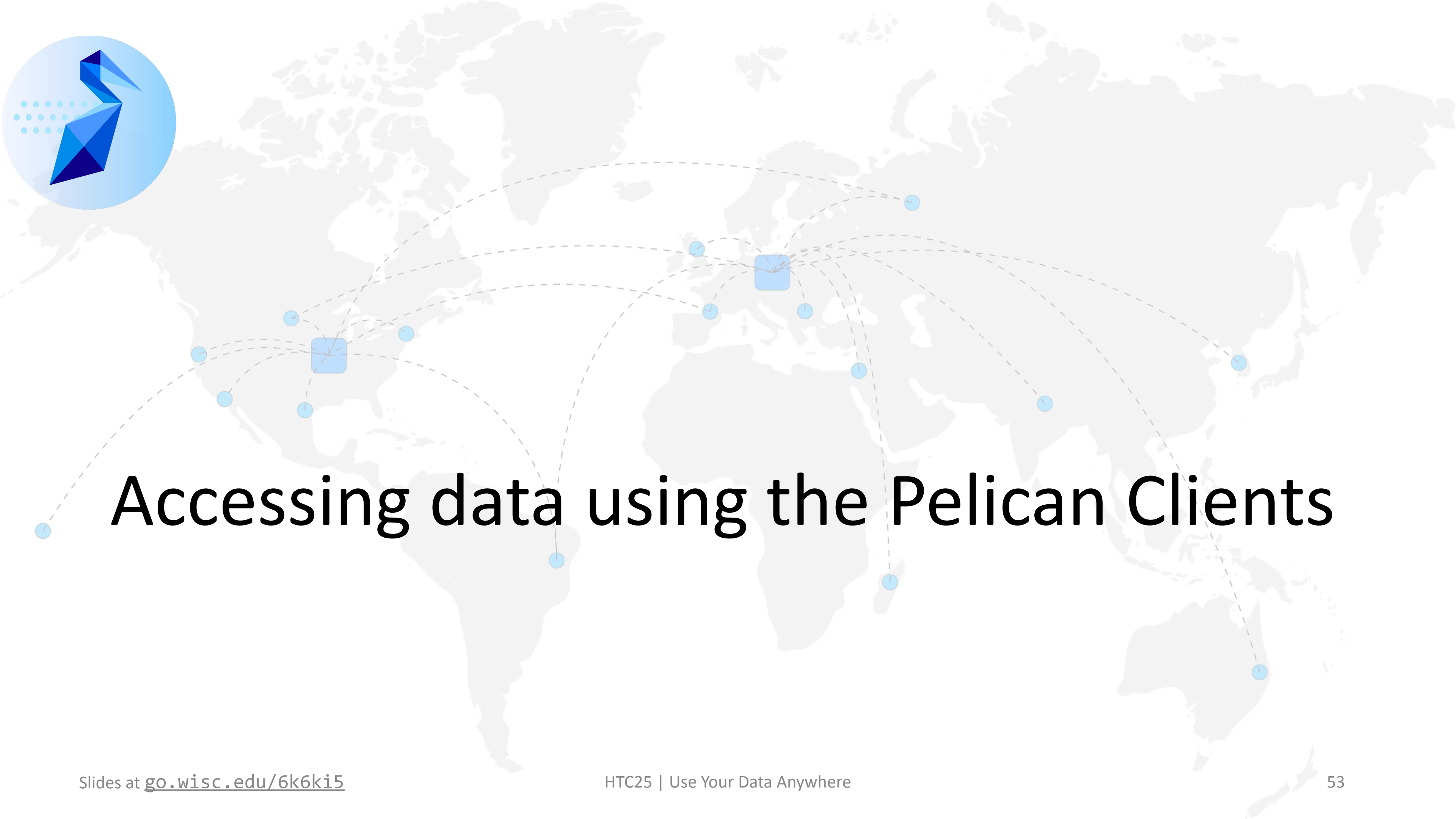


🎉Done! 🎉

The importance of caching

- **Pelican assumes that objects *do not change!***
- Requirements for accessing objects are respected **everywhere**
- Caches prevent the data repository from being overwhelmed

The OSDF maintains over **30 caches** across North America and Europe.
Most are high-performance servers on internet backbones.
A major advantage for connecting your data to the OSDF!



Accessing data using the Pelican Clients

Pelican Clients

Pelican provides several Clients for interacting with Pelican Federations

- Pelican CLI
- PelicanFS
- HTCondor Plugin (built-in to HTCondor)

Pelican Clients

Pelican CLI

- Command Line Interface
- Available as a standalone binary

PelicanFS

- Python Interface, built on `fsspec`
- Available with pip/conda

HTCondor Plugin

- Can use Pelican/OSDF URLs in your submit files for HTCondor jobs
- Built-in to HTCondor

Basic Client actions

`ls`

- List the names of objects accessible via a namespace of a Pelican Federation

`get`

- Download a copy of an object via a Pelican Federation

`put`*

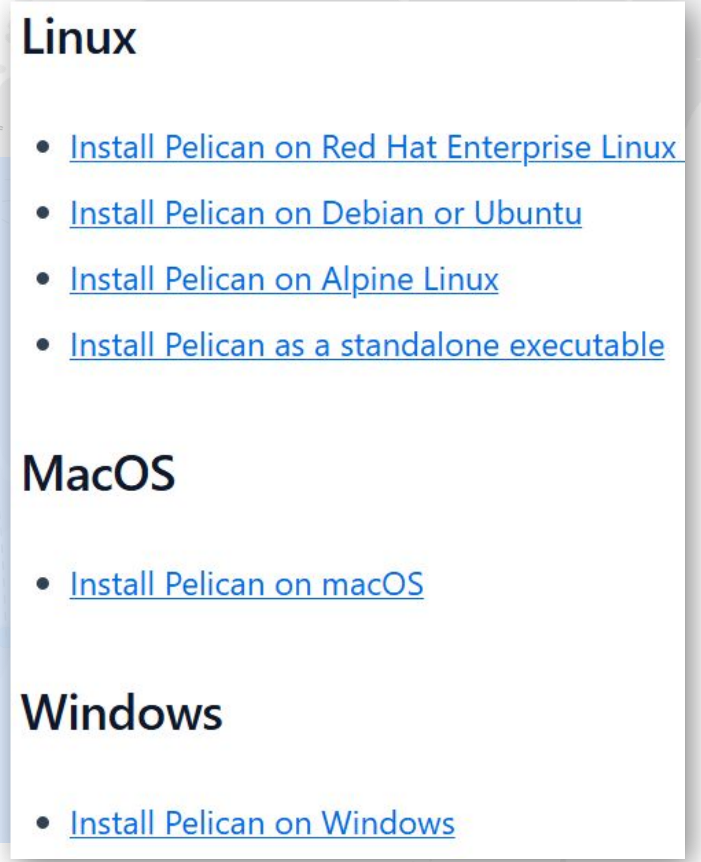
- Upload an object to make it available within a namespace of a Pelican Federation

*always requires authentication - not covered today

Pelican CLI

Standalone binary - single file, does not require admin permissions!

docs.pelicanplatform.org/install



Linux

- Install Pelican on Red Hat Enterprise Linux
- Install Pelican on Debian or Ubuntu
- Install Pelican on Alpine Linux
- Install Pelican as a standalone executable

MacOS

- Install Pelican on macOS

Windows

- Install Pelican on Windows

Pelican CLI

Once installed, available as the `pelican` command.
Uses noun-verb syntax.

Client commands use the `object` noun:

```
pelican object <request> <additional arguments>
```

(Additional arguments usually involves a Pelican URL)

Pelican CLI

Listing objects

Use this command to list the object(s) accessible via a Pelican URL:

```
pelican object ls <Pelican URL>
```

For example,

```
pelican object ls osdf:///pelicanplatform/test
```

Pelican CLI

Listing objects

Use this command to list the object(s) accessible via a Pelican URL:

```
pelican object ls <Pelican URL>
```

Remember,

`pelican://osg-htc.org/______________________________`

is exactly the same as

`osdf:///______________________________`

Pelican CLI

Listing objects

Use this command to list the object(s) accessible via a Pelican URL:

```
pelican object ls <Pelican URL>
```

For example,

```
pelican object ls osdf:///pelicanplatform/test
```

Pelican CLI

Listing objects

For more information, use the `-l`/`--long` flag:

```
pelican object ls --long <Pelican URL>
```

For example,

```
pelican object ls --long osdf:///pelicanplatform/test
```

Pelican CLI

Getting objects

To get an object via its Pelican URL, use the command

```
pelican object get <Pelican URL> <destination>
```

For example,

```
pelican object get
osdf:///pelicanplatform/test/hello-world.txt ./
```

Pelican CLI

Getting objects recursively

To get all of the objects associated with a specific namespace, you can use a "query", specifically, the `?recursive` query:

```
pelican object get <Pelican URL>?recursive <destination>
```

For example,

```
pelican object get
osdf:///pelicanplatform/test?recursive ./
```

A peak behind the curtain…

Using '--debug'

Verbose information about the Pelican Client can be reported using the `--debug` flag.

For example,

```
pelican object get --debug osdf:///pelicanplatform/test/hello-world.txt
```

PelicanFS

For a Python-based Client, use Pelican's `pelicanfs` Python package

Install it with `pip` like other Python packages:

```
python3 -m pip install pelicanfs
```

This provides a Pythonic interface to interact with a Pelican Federation, usable in scripts and the Python console

PelicanFS

Import and setup

Before proceeding, move into the

`pelican-training-client`

directory!!

PelicanFS

if time allows

Import and setup

First, launch the `python3` terminal console by entering

```
python3
```

PelicanFS

if time allows

Import and setup

The `pelicanfs` client provides the `PelicanFileSystem` class, based on the `fsspec` package.

Next, import the necessary class:

```
from pelicanfs.core import PelicanFileSystem
```

PelicanFS

if time allows

Import and setup

The `pelicanfs` client provides the `PelicanFileSystem` class, based on the `fsspec` package.

Now instantiate an instance of the object, defining the Discovery URL when you do so:

```
    pelfs = PelicanFileSystem('pelican://osg-htc.org')
```

PelicanFS

if time allows

Client Methods

Now instantiate an instance of the object, defining the Discovery URL when you do so:

```
pelfs = PelicanFileSystem('pelican://osg-htc.org')
```

The methods of this object provide are the mechanisms of the Client and take the form

```
pelfs.<method>(<arguments>)
```

where `<arguments>` usually contains the rest of the Pelican URL

PelicanFS

if time allows

Listing objects

List objects by providing the namespace to the `ls` method:

```
pelfs.ls('/namespace')
```

The result will be a list of dictionaries. For example,

```
list_results = pelfs.ls('/pelicanplatform/test')
```

To see just the names of the objects, run

```
[result['name'] for result in list_results]
```

PelicanFS

if time allows

Getting objects

Use the `get` or `get_file` method to get an object:

```
pelfs.get_file('/namespace/object_name', 'destination')
```

For example,

```
pelfs.get_file('/pelicanplatform/test/hello-world.txt',
                   'hello-world.txt')
```

(`get` is currently bugged..?)

PelicanFS

if time allows

Getting objects

Then use the file the typical Python way:

```
with open('hello-world.txt', 'r') as f:
    my_file = f.read()

print(my_file)
```

PelicanFS

if time allows

Getting objects the `fsspec` way

Instead of manually copying the object to the local file system, let `fsspec` do it for you:

```
with pelfs.open('/pelicanplatform/test/hello-world.txt', 'r') as f:
    direct_read = f.read()

print(direct_read)
```

PelicanFS

Automatically (!) get objects

Lots of Python packages automatically use `fsspec` behind the scenes for data transfers. The same is true for `pelicanfs`!!

To demonstrate this, exit the Python console:

```
exit()
```

PelicanFS

Automatically (!) get objects

Take a look at the included script:

```
cat autoload.py
```

Note that `pelicanfs` is not referenced anywhere in the script!

PelicanFS

Automatically (!) get objects

Now execute the script:

```
python3 autoload.py
```

After a minute or two, you'll see the dataset!

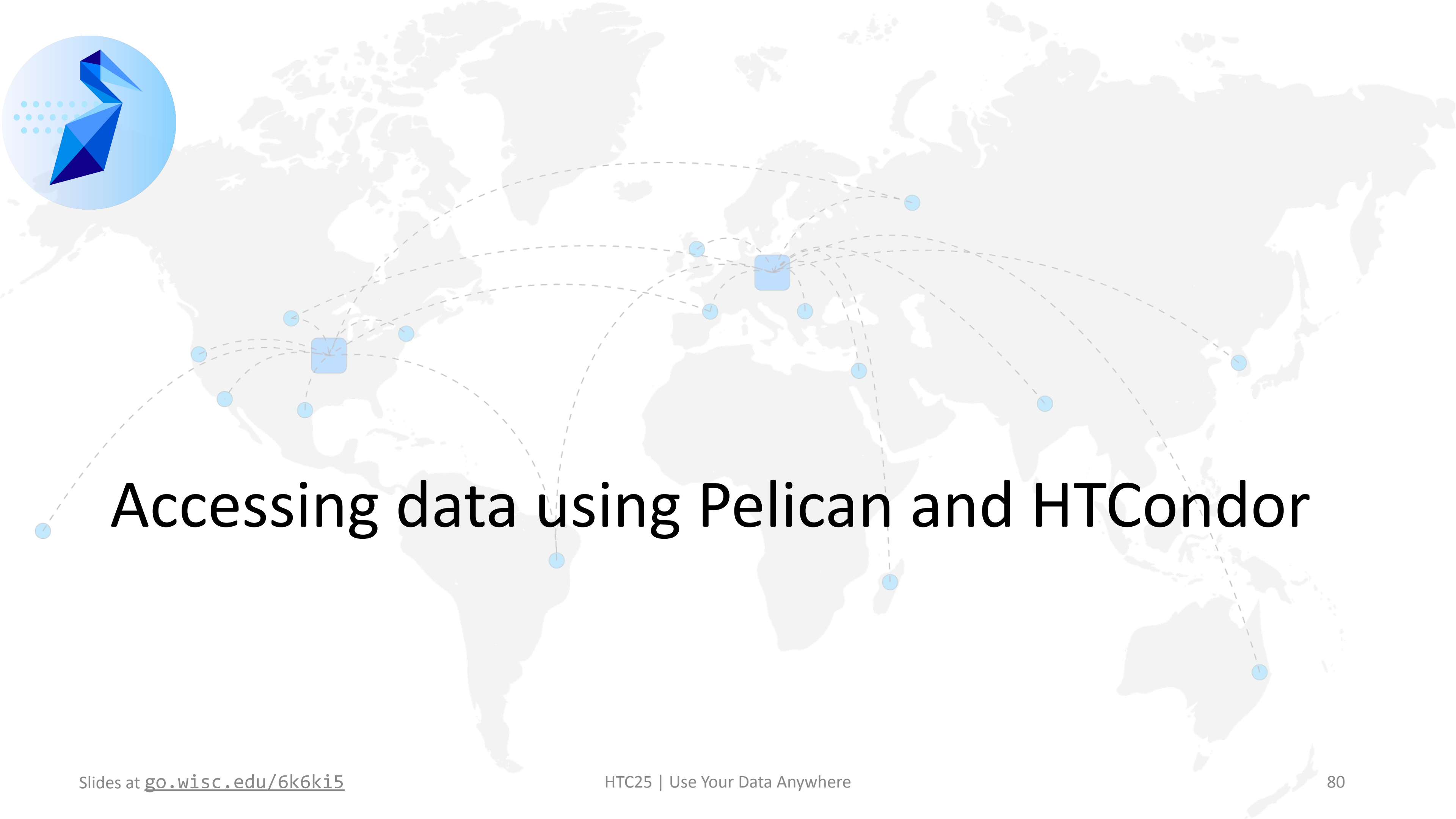
Behind the scenes, `pandas` knows to use the `pelicanfs` package to download data with `osdf://` and `pelican://` URLs

PelicanFS

Notes about `pelicanfs`:

- Not all features of `fsspec` are available (since the Pelican Client is not a proper filesystem..)
- Still in development!

Accessing data using Pelican and HTCondor



Pelican Plugin

Pelican has tight integration with HTCondor

Among other things, **can use Pelican URLs in your submit file**!

- Normal declaration, but using `pelican://` or `osdf://` file transfer protocol
- HTCondor manages the transfer
- Errors are automatically retried, or turned into holds

Pelican Plugin

Transferring as input

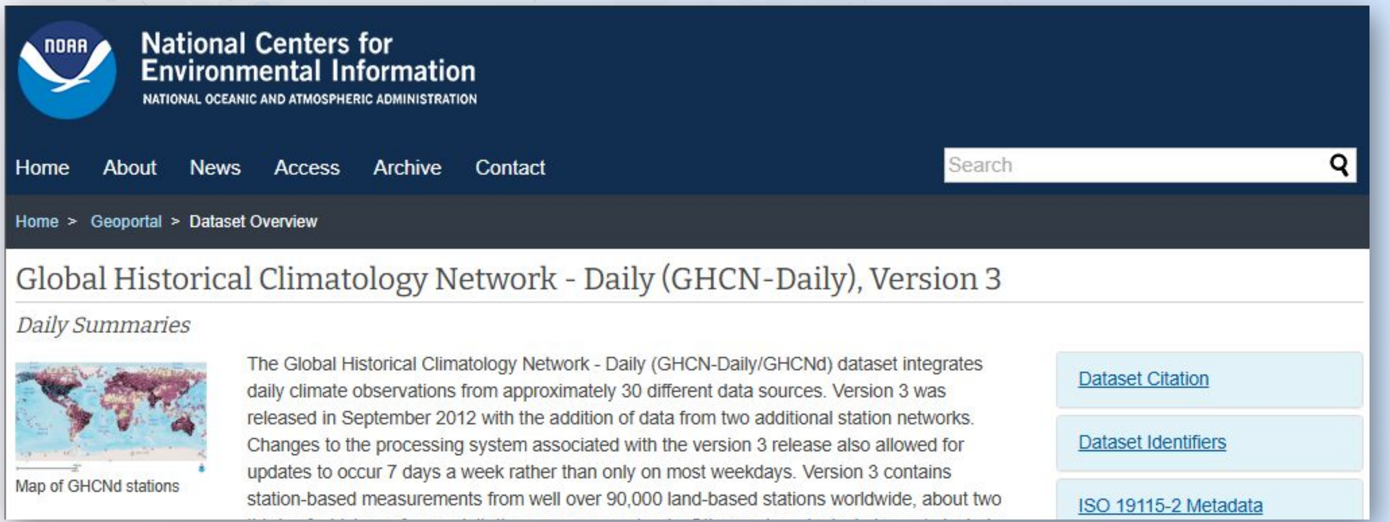
In the submit file, inputs to be transferred are declared using `transfer_input_files`. Just put the Pelican URL in that list for the object you want to transfer as input.

For example,

```
transfer_input_files =
osdf:///pelicanplatform/test/hello-world.txt
```

Data & Compute at Scale

To demonstrate the power of Pelican's HTCondor Plugin, let's work on a small exercise using climate data from NOAA



National Centers for Environmental Information
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
Home About News Access Archive Contact
Search
Home > Geoportal > Dataset Overview

Global Historical Climatology Network - Daily (GHCN-Daily), Version 3

Daily Summaries

Map of GHCNd stations

The Global Historical Climatology Network - Daily (GHCN-Daily/GHCNd) dataset integrates daily climate observations from approximately 30 different data sources. Version 3 was released in September 2012 with the addition of data from two additional station networks. Changes to the processing system associated with the version 3 release also allowed for updates to occur 7 days a week rather than only on most weekdays. Version 3 contains station-based measurements from well over 90,000 land-based stations worldwide, about two

The climate dataset

The GHCN dataset from NOAA is available via the OSDF!

- Part of Amazon's Open Data repository (existing namespace at `/aws-opendata`)
- Connected via US East 1 (`us-east-1`)
- S3 domain name is `noaa-ghcn-pds`

Altogether, the data should be accessible via

```
osdf:///aws-opendata/us-east-1/noaa-ghcn-pds
```

(but currently can't `ls` that namespace..)

The climate dataset

Structure looks like this (ignoring lots of other files..)

```
osdf:///aws-opendata/us-east-1/noaa-ghcn-pds/
    ghcnd-stations.txt
    csv/
        by_station/
            <STATION_ID_1>.csv
            <STATION_ID_2>.csv
```

The climate dataset

Move into the `htcondor-plugin` directory.

Then get the stations list using one of the Clients.

Recommend copying the `pelican object get` command from the Github README (under the "Exploring the data" subheading)

The climate "analysis"

Download a csv file for a station using Pelican.

Example command under "A rudimentary climate analysis" heading.

Then run the `example.py` script with the station ID as an argument.

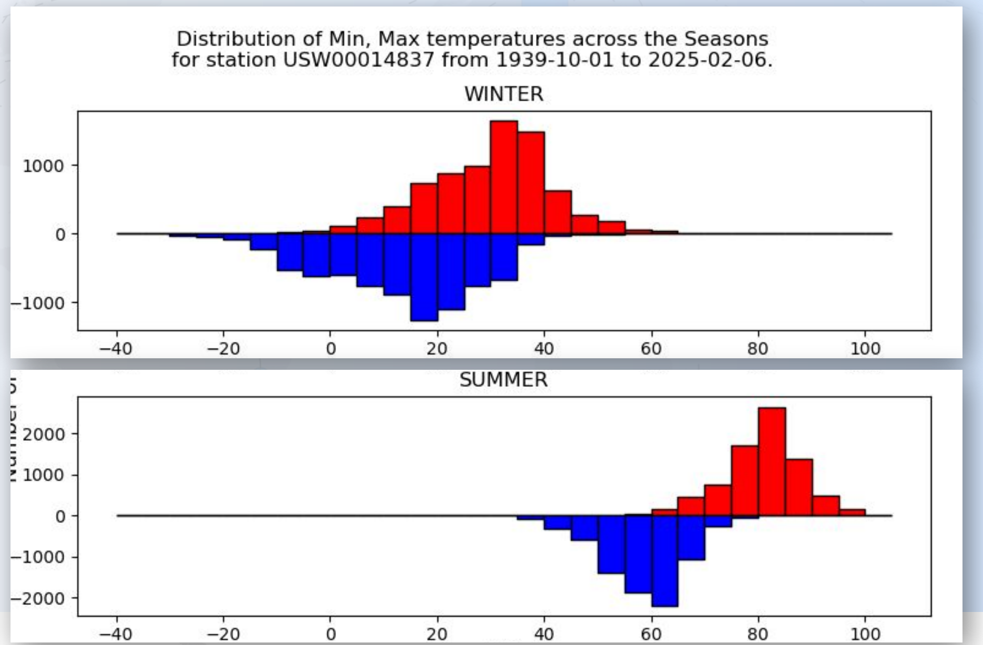
For example,

```
./example.py USW00014837
```

This will create a `.png` file!

The climate "analysis"

Winter is colder than summer! (at least in Wisconsin)



Scaling out

Suppose you want to analyze ALL the stations

- Dataset contains ~130,000 stations!!

Suppose you also want to do a better analysis, but it takes 1 hour per station

- Running all stations in serial (one after the other) will take …

$$130{,}000 \text{ stations} \times \frac{1 \text{ hour}}{1 \text{ station}} = \mathbf{15\ years}$$

💀💀💀

Scaling out with HTCondor

High Throughput Computing (and HTCondor) to the rescue!

- On a system like the OSPool, can easily run 1000s of jobs at a time

$$130{,}000 \text{ stations} \times \frac{1 \text{ hour}}{1{,}000 \text{ stations}} = 130 \text{ hours} \approx \mathbf{5\ days}$$

What about the data movement?

Scaling out with HTCondor and Pelican

Pelican (and the OSDF) to the rescue!

- You don't need to manually stage the dataset - just give HTCondor the correct Pelican URLs

For example,

```
OSDF_PREFIX =
osdf:///aws-opendata/us-east-1/noaa-ghcn-pds/csv/by_station/
transfer_input_files = $(OSDF_PREFIX)/$(STATION_ID).csv
queue STATION_ID from station_list.txt
```

Submit a list of climate analysis jobs

In the `htcondor-plugin` directory, take a look at the example submit file:

```
cat example.sub
```

Generate the list of stations to analyze using the script:

```
./generate_list.sh
```

Submit a list of climate analysis jobs

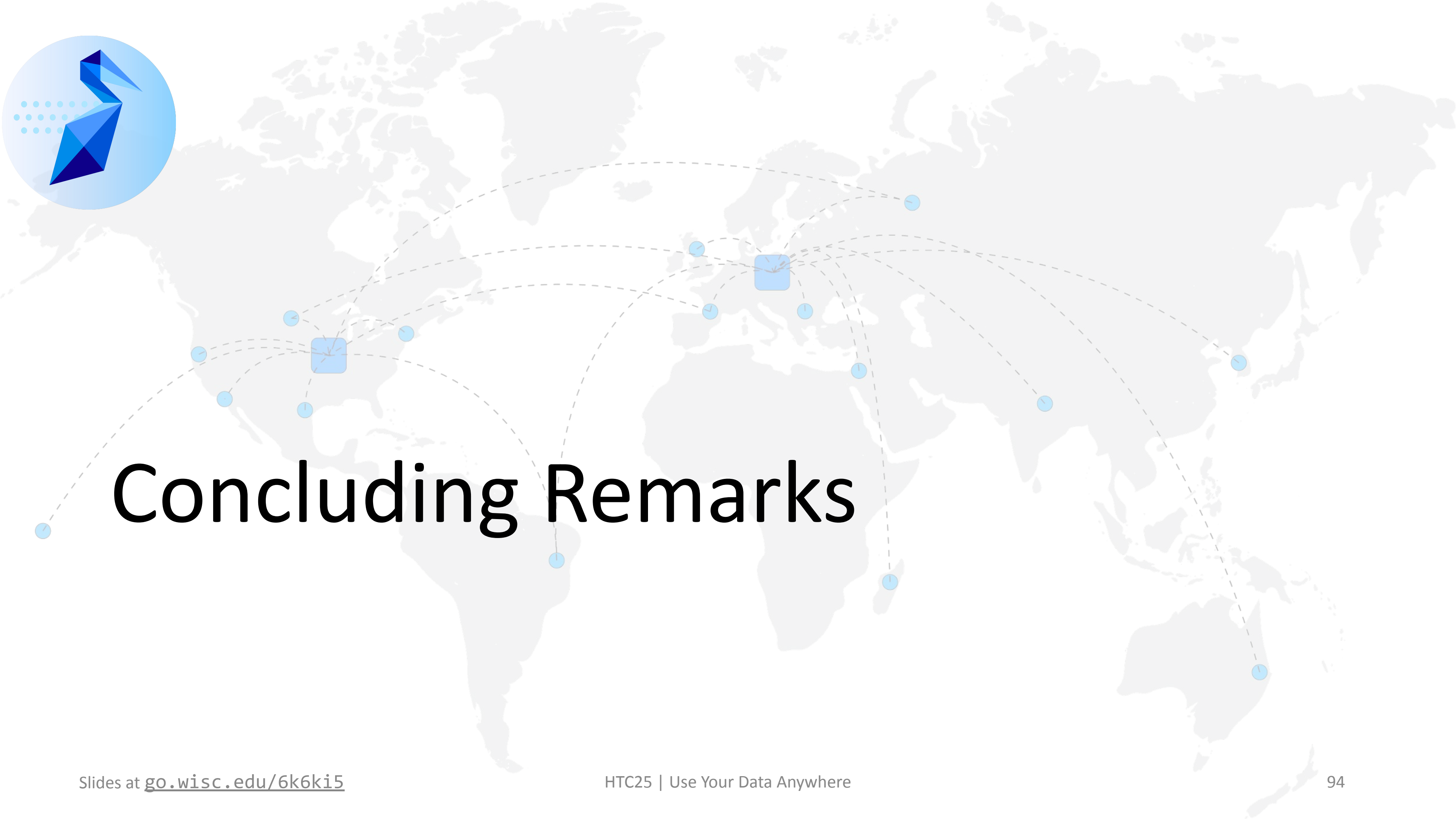
Submit the list of jobs with

```
condor_submit example.sub
```

Monitor the progress of the jobs with

```
condor_watch_q
```

If you really wanted to do a complex analysis of all 130,000 stations, please don't use the Guest OSPool Notebook to do so!
Instead, request a full OSPool account at portal.osg-htc.org/application



Concluding Remarks

Next Steps

More information on Pelican:

pelicanplatform.org

docs.pelicanplatform.org

More information about the OSDF:

osg-htc.org/services/osdf

osg-htc.org/services/osdf/data

Trainings and Getting Help

PEARC25! Longer version of this tutorial that will also cover authentication

For OSDF support, email: support@osg-htc.org

For Pelican support, email: help@pelicanplatform.org

Questions?

Acknowledgements

This material is based upon work supported by the National Science Foundation under Cooperative Agreements OAC-2331480. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.