Welcome!

Wifi: PCOC_Visitor

Access Code: SSA2026 *(not case sensitive)*

While you are waiting for the workshop to start, please go to earthscope.org/workshop and proceed with the ***Before We Start*** steps:

- Ensure you have a working EarthScope account that has been enabled for GeoLab access
- Launch GeoLab, select the **29GB RAM / 4CPU resource option**
- Clone or update the git repository

Advancing Geophysical Research with Cloud Computing

Seismological Society of America
Annual Meeting 2026
Pasadena, CA

NATIONAL GEOPHYSICAL FACILITY
Operated by EarthScope

Workshop Outline

Introduction to EarthScope

Brief update on cloud migration status

Cloud-friendly Concepts Overview

Primary Learning Objectives

Running Notebooks in GeoLab

About Us

https://www.earthscope.org/about/earthscope/

NSF | NGF Operated by EarthScope

data stewardship

cost-effective computing

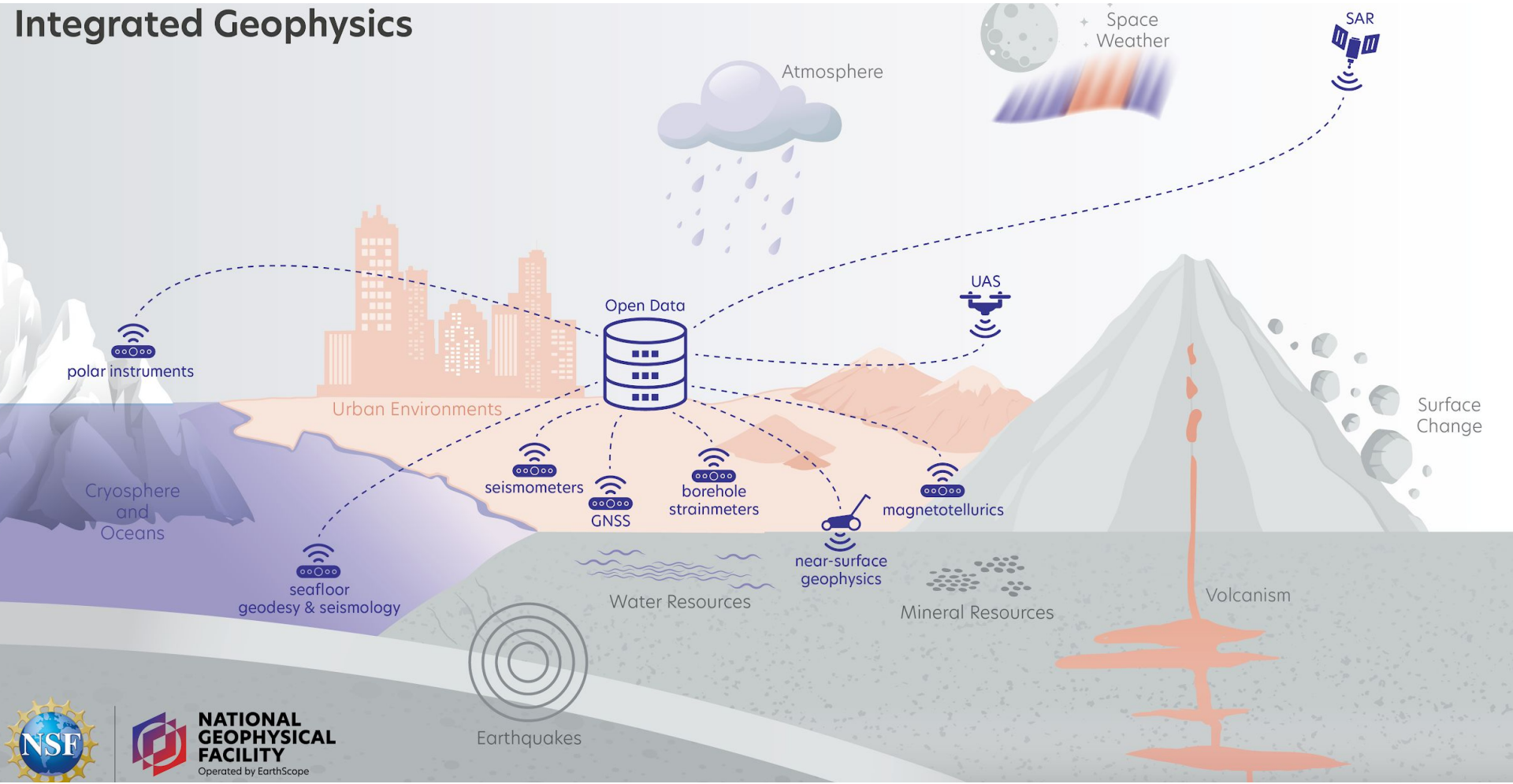
democratized access

IRIS

UNAVCO

EarthScope Consortium

Integrated Geophysics



NATIONAL GEOPHYSICAL FACILITY
Operated by EarthScope

On-Ramp Group & Course Instructors

Cross-directorate, multi-domain group

Sophia

Tammy

Scott

Chad

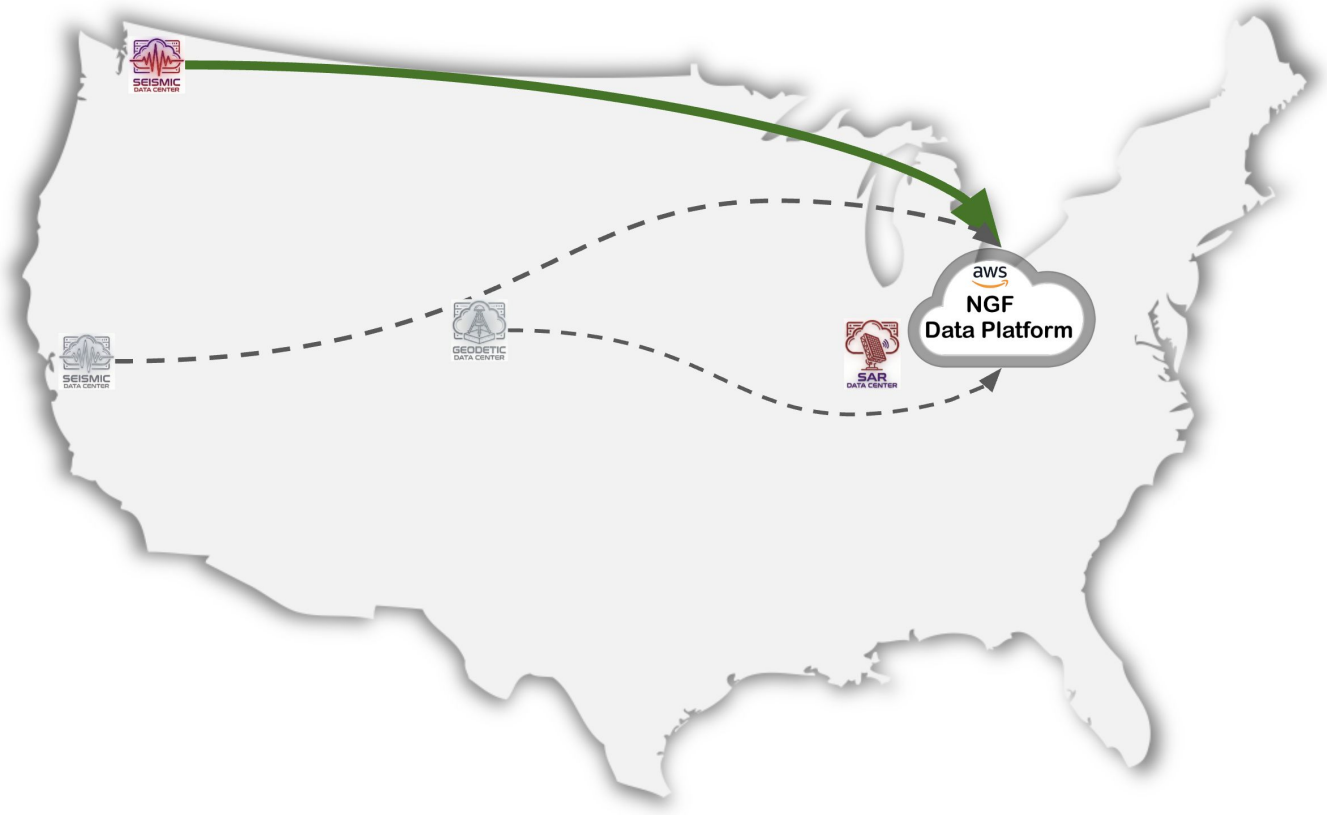
Michael

Sarah

Robert

Justin

Cloud Migration Path



What does the migration mean for you?

Identity Management / EarthScope Accounts
(www.earthscope.org/user)

New service URLs
(service.iris.edu → service.earthscope.org)

Soon: S3 object store access

Workshop Overview

What are we doing today?

- Identify opportunities to adapt your existing work to cloud-friendly models and methods
- Progress your own cloud literacy path
- Focus will be how working in cloud-based architecture can support your data analysis and workflows.

Our Example Workflow Overview

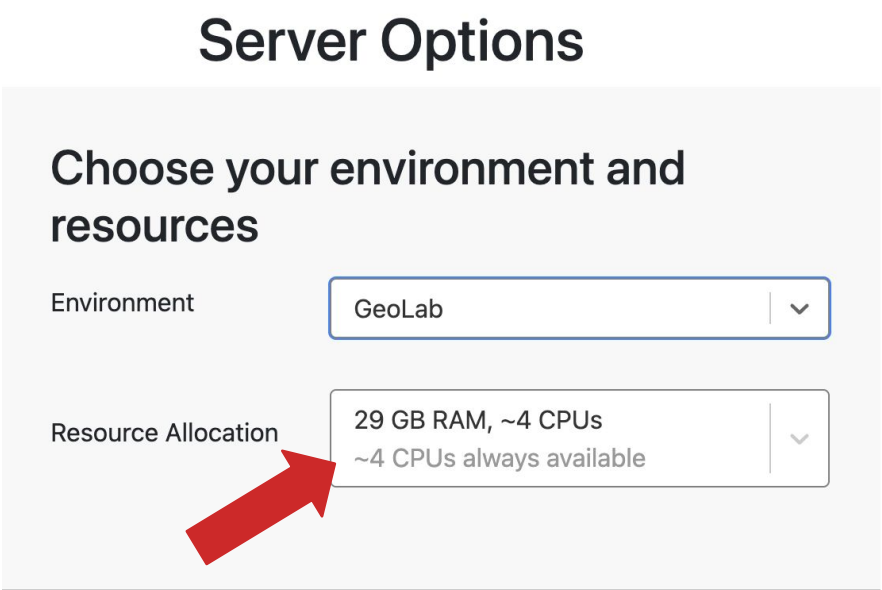
1. Environment Selection + Resource Allocation
2. Data Download
   a. The slow way (serially)
   b. The fast way (concurrently)
3. Data Pre-processing
   a. Parallel computing
4. Putting 2&3 together: efficient computing with in-stream processing
5. Example: Matched Filtering Cross Correlation
6. Saving data and work

Primary Concepts/Take Aways

- GeoLab is an extensible and purpose built environment for geophysical research
- Jupyter notebooks facilitate shareable and reproducible research
- GeoLab is great for exploratory data analysis
- Notebook code reuse is supported by exporting it to standard python for use in production
- Encourage the use of best Python programming practices

Selecting the Right Resources

https://www.earthscope.org/data/geolab/



Environment (software):

- Pre-configured set of software that is installed at launch
- Consistent + reproducible

Resource Allocation (hardware):

- What size machine (or machine partition) you're working on
- Shared servers guarantee you a minimum amount of compute availability
  - You may have access to more resources during periods of lower traffic.
  - Shared resources reduce operating costs!
- Always start small and watch your resource utilization before sizing up!
- Larger servers and GPU instances are available by request

earthscope.org/workshop *Step 2-3*

Pull / Update The Repository

Recommended approach: create a new copy of the repository

```
cd                          #go home
mkdir ssa-2026              #make a new file directory
cd ssa-2026                 #switch into that directory
git clone https://github.com/EarthScope/GeoLab-learning-hub.git
cd GeoLab-learning-hub/workshops/ssa_workshop_2026/
```

Alternative approach if you already have /Geolab-learning-hub in your file directory:

```
cd Geolab-learning-hub/workshops/ssa_workshop_2026
git stash       #stash any local changes
git pull        #get the latest upstream version
git stash pop   #(optional) restore your local changes
```

earthscope.org/workshop *Step 4-5*

Environment Basics:

- The base GeoLab Environment contains many commonly used data science and geophysics software packages
  - Check for your favorite package: `conda list | grep obspy`
- Use a package manager like **conda**, **pip,** or **mamba** to install additional packages
  - Ephemeral installations do *not* persist from session to session
  - Put these in your notebook with a “magic” command
    - `%conda install -c conda-forge <pkg-name> --yes`
    - `%pip install <pkg-name>`

Workshop Outline

Introduction to EarthScope

Brief update on cloud migration status

Cloud-friendly Concepts Overview

Primary Learning Objectives

Running Notebooks in GeoLab

iMUSH (2014-2016)

Imaging Magma Under St. Helens

- Multi-domain investigation of Mt St Helens magma system

- Dense nodal array monitoring passive & active seismic sources
  - 900+ nodal seismographs around summit
  - Large-N (2,000+) array of single-channel Reftek recording active sources
  - **70 3-component broadband stations deployed continuously for 2-years**

- Phased deployment of MT array

25+ publications, with more to come as data access and AI/ML techniques improve

Data Access Basics

data_shipment

my_awesome_request

verify

authenticate

locate

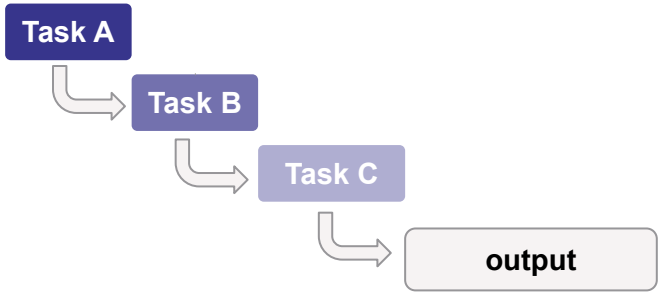
organize

```
# 1_efficient_data_access.ipynb
```

Sequencing vs. Parallelization

Sequential

Tasks are executed one after another in a linear timeline.

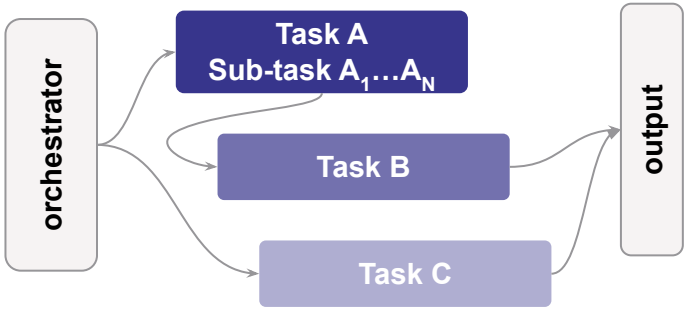


Building a house with one worker. Each task is completed before the next one starts.

- Strict order of operations
- Tasks must run to completion

Parallel

Tasks overlap in time, effectively utilizing system resources.



Building a house with a project manager and many workers who are task-switching.

- Sub-tasks are simultaneous
- Starting one task does not depend on another to finish

Parallelization Strategies

Concurrency

- Use rapid task switching between multiple threads on the same CPU core
- Make use of idle / wait-time
- *Pass the hammer to another worker while waiting for a job to complete.*

Best For: I/O-bound tasks where the bottleneck is waiting for external resources

Implementation:

```
ThreadPoolExecutor()
```

MultiProcessing

- Execute multiple independent tasks by spinning up multiple python instances on different CPU cores
- *Hire more workers, each worker gets their own hammer.*

Best For: CPU-bound tasks where the bottleneck is computational demand

Implementation:

```
ProcessPoolExecutor()
```

Concurrency vs MultiProcessing

Discussion with your neighbor:

- Which is the better strategy for downloading many files?

- Which is the better strategy for applying filters and data pre-processing operation to large datasets?

Parallelization, Applied

Sequential

One task finishes completely before the next one is started

Best for prototyping and tinkering

- Phase pickers → associators → locators → event detection
- First-arrival onsets → determine uncertainty → invert for model

Concurrent

While one task is in 'idle' or 'wait' status, the worker starts up the next task

Best for I/O-bound tasks, where tasking the database is limited

- Data-download or file read/write steps

Multi-processed

Two tasks are completed by different workers, independently of each other

Best for CPU-bound tasks, like computational pre-processing on many independent files

- Phase picking 1000 channels
- 3-D seismic surveys (prestack migration + layered attentuation)

Parallelization Recap:

- Good for processing large amounts of independent data
- Concurrency: for I/O bound tasks like file downloads
- MultiProcessing: for CPU-bound tasks like computational processing

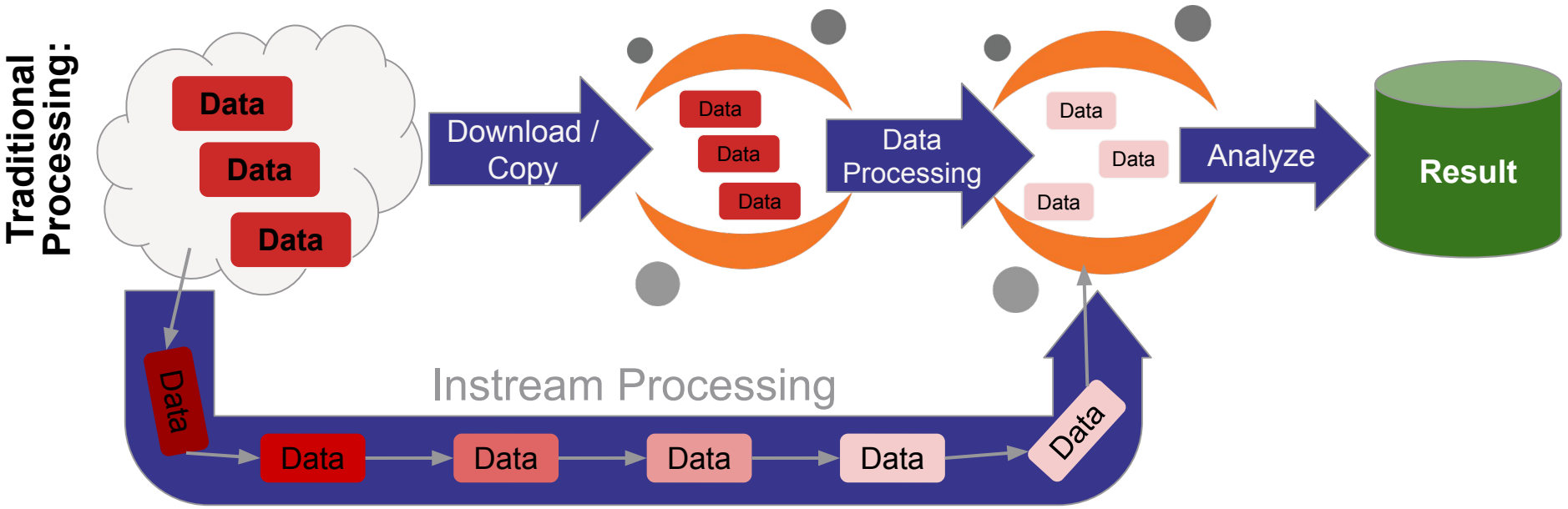
Discussion Questions:

- At what data volume threshold, or file organization system, does it make sense to apply a parallel approach?
- Which science workflows or workflow steps are well-suited to parallelization? Which ones aren't?

What other scaling solutions are available in the cloud?

Parallelization (Horizontal Scaling)	• python concurrent.futures module • Dask • MsPASS
Server Size (Vertical Scaling)	• RAM + CPU • Larger servers available on request: help@earthscope.org
GPU (Machine Learning & Image Processing)	• Available on request: help@earthscope.org

What is InStream Processing?



Transform data while in active memory (faster)

Only store processed data (reduced storage)

No need for intermediate file management

Always get latest copy from FDSN server

Improved scalability for big datasets

Which approach is right for me?

Instream

- Processing many stations with limited disk space
- Need only processed data, not raw data
- Working with cloud computing (minimize compute and storage costs)

IMPLEMENTATION:
obspy streams

Batch/Download First

- Need to preserve raw data for multiple processing runs
- Working offline or with unreliable network
- Need to experiment with different processing parameters (prototyping)
- Archiving data for long-term studies

IMPLEMENTATION:
obspy Mass Downloader

```
# 3_in_stream_processing.ipynb
```

Customizing Environments

Basic: Use “magic commands” in a notebook to install packages

```
%conda install -c conda-forge <pkg-name> --yes
```

Intermediate: Create a conda environment

- specify a list of python packages to install
- IN GEOLAB: install the conda environment as a python kernel and activate it

```
%conda create  -n <env-name> -c conda-forge
<pkg-names>
!python -m ipykernel install --user --name=<env-name>
```

Complex: Build a custom image

Write a custom dockerfile that can include more complex software installations

 # 4_cross_correlation.ipynb

Matched filtering cross correlation

- This seismic example demonstrates how to get a list of events and manipulate the data in pandas for analysis, i.e., find the dates with the most events
- Demonstrates how to select events using pandas
- Create an inventory of stations with high frequency vertical channels and create picks from the catalog of events for each station **(note: add an exercise to parallelize this task)**
- Use eqcorrscan to create templates to find P waves
- Use the templates to find new events - **looking for suggestions if we should do more analysis, e.g., stacking, etc**

```
# 4_cross_correlation.ipynb
```

Saving + Sharing Work

- Reproducible Conda Environments
- Downloading files out of GeoLab
- Saving to GitHub

```
# 5_saving_work.ipynb
```

Environment Management

- `environment.yml` is a list of packages + versions
  - Can be replicated on another computer (cloud or local)
- Exporting an environment from GeoLab:
  - Captures *all* packages in the base GeoLab environment, *plus* any ephemeral installs
  - Re-installing in GeoLab can be time consuming due to re-installation over existing environment
- We are working on *simplifying* the base GeoLab environment
  - Reduce packages and dependency conflicts
  - Faster server launch times

`# 5_saving_work.ipynb`

Working in Git

- A **tool** for version control, collaboration, publication
- Widely used in software development and code management
- Allows a user or group to:
  - Make and test changes on an independent branch without disrupting work for other users
  - Tracked changes/snapshots at fixed waypoints (commits)
  - Auto-managed static version numbers (commit hashes)
- Requires making an account with a service provider
  - **GitHub** - popular with open source projects, research groups
  - **GitLab** - popular with proprietary & industry organizations

`# 5_saving_work.ipynb`

What Did We Do Today?

1. Used GeoLab as an introductory cloud environment
   a. Learned two strategies to customize the environment (ephemeral installations, conda environments)
   b. Considered different resource sizing considerations
2. Explored strategies for streamlined cloud data access
   a. In-stream processing
3. Compared parallelization approaches
   a. Concurrency, multi-processing

QUESTIONS?

Survey

Thank you for participating in the SSA Cloud Computing Workshop.

Please complete our short workshop survey.

Also linked on earthscope.org/workshop

https://irisepo.iad1.qualtrics.com/jfe/form/SV_d7u7wRz5wihfC0C