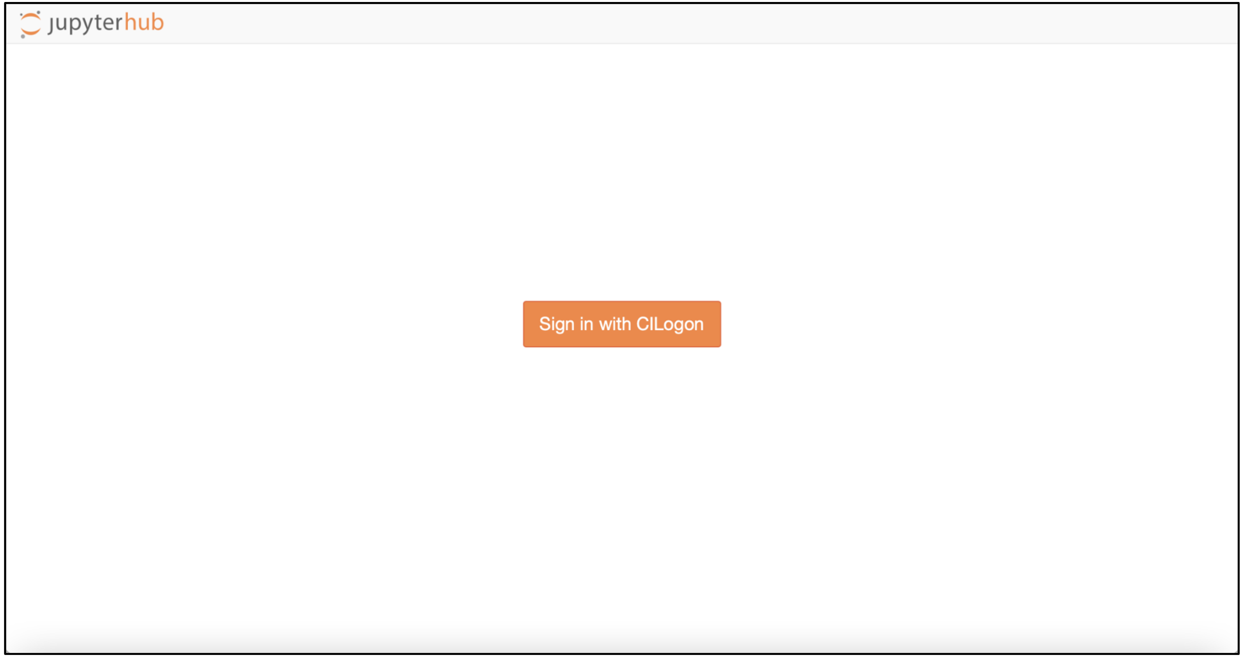
Building a Workflow

Christina Koch
Throughput Computing 2023
July 12, 2023

Open an internet browser and enter:

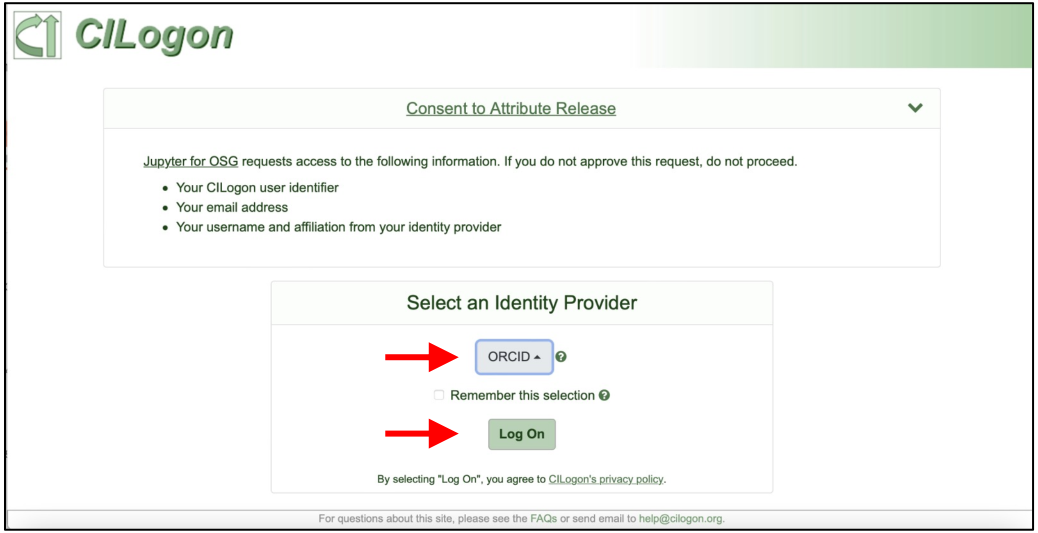
https://notebook.ospool.osg-htc.org



Log into an OSPool Access Point

Login using any of the available authentication options. Some choices:

- NIH login
- Google (i.e. gmail)
- GitHub
- ORCID



Consent to Attribute Release

Jupyter for OSG requests access to the following information. If you do not approve this request, do not proceed.

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

Select an Identity Provider

ORCID

□ Remember this selection

Log On

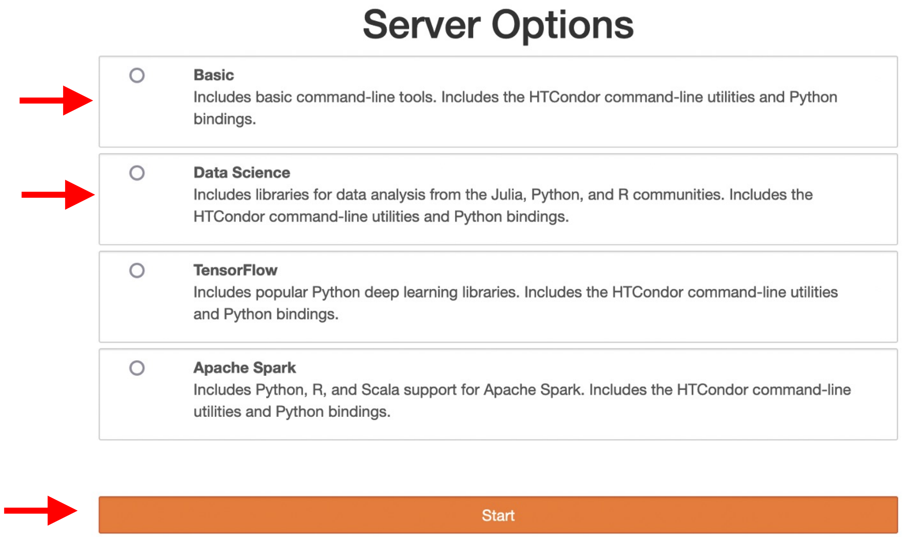
By selecting "Log On", you agree to CILogon's privacy policy.

For questions about this site, please see the FAQs or send email to help@cilogon.org.

Launch Data Sciences Notebook

1. Click the “**Data Science**” or “**Basic**” box

2. Click orange “**Start**” button



Server Options

○ **Basic**
Includes basic command-line tools. Includes the HTCondor command-line utilities and Python bindings.

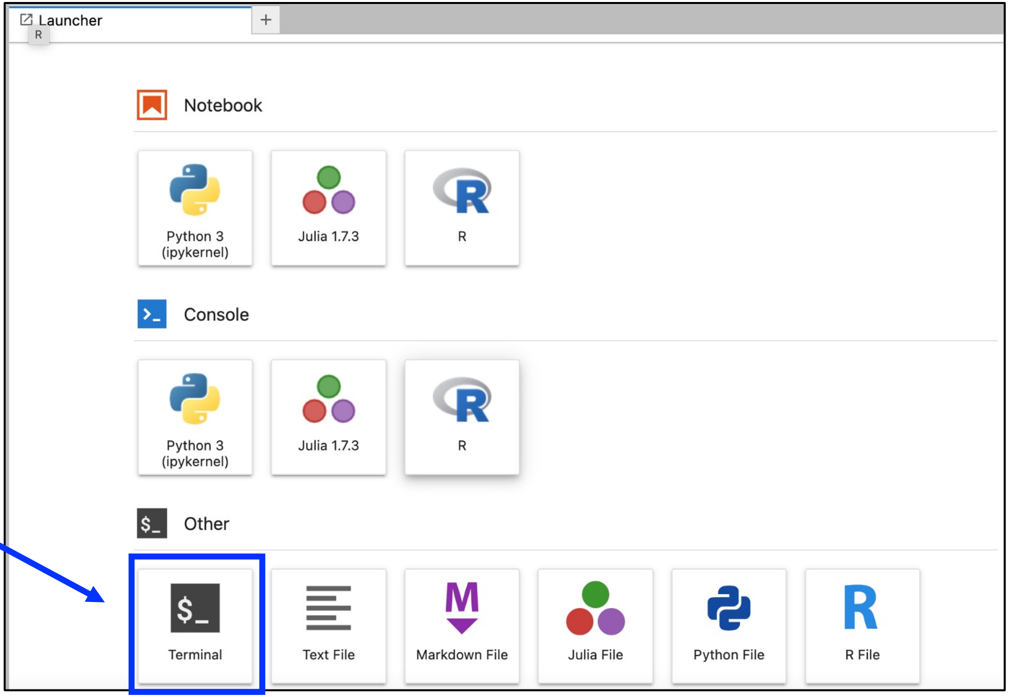
○ **Data Science**
Includes libraries for data analysis from the Julia, Python, and R communities. Includes the HTCondor command-line utilities and Python bindings.

○ **TensorFlow**
Includes popular Python deep learning libraries. Includes the HTCondor command-line utilities and Python bindings.

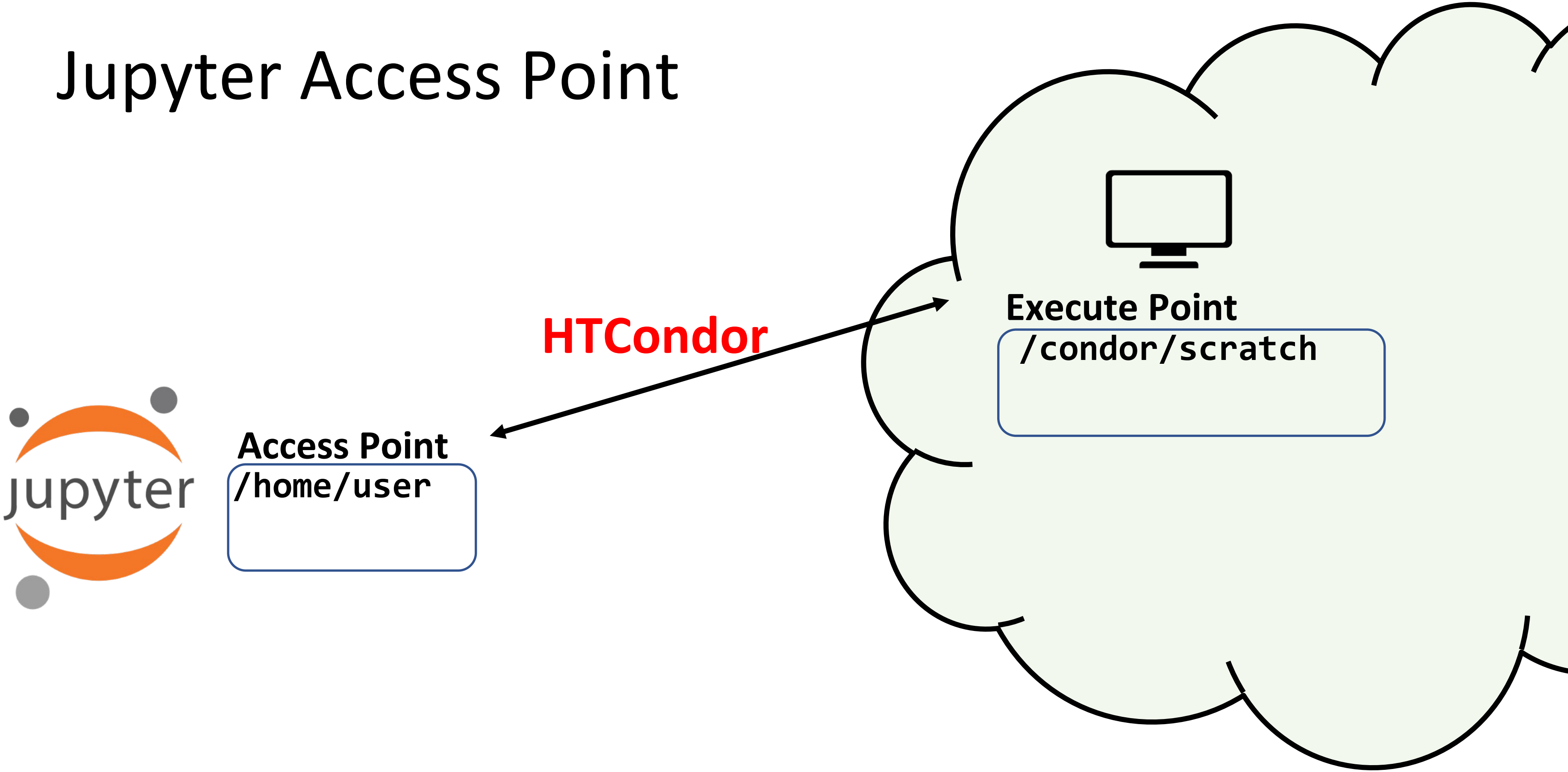
○ **Apache Spark**
Includes Python, R, and Scala support for Apache Spark. Includes the HTCondor command-line utilities and Python bindings.

Log into an OSPool Access Point

Open a Terminal

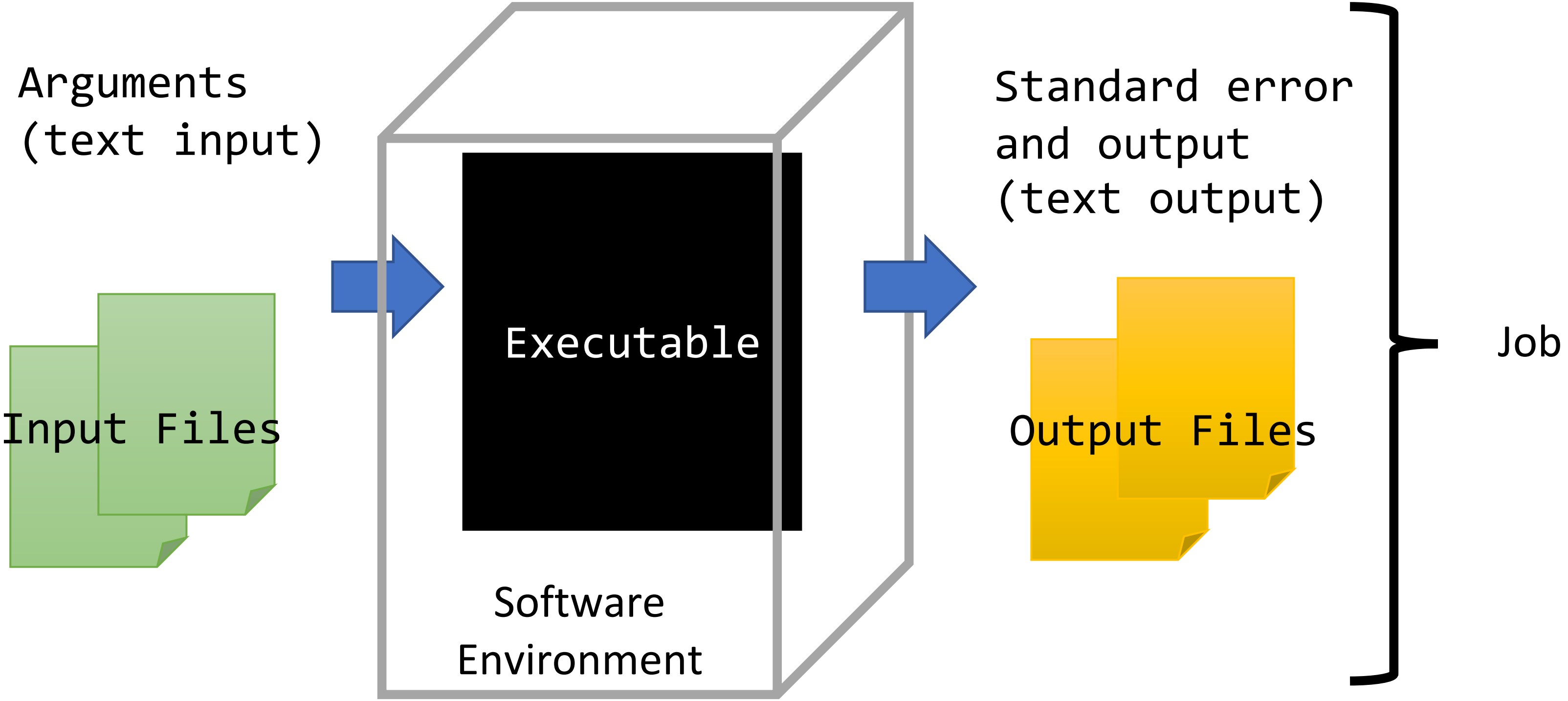


Jupyter Access Point



Use Cases

Job Component Vocabulary


Arguments
(text input)

Input Files

Executable

Software
Environment

Standard error
and output
(text output)

Output Files

Job

Analyzing Multiple Files

- **Software**: bwa aligner
- **Executable**: Shell script with bwa commands
- **Arguments**: None (for now…)
- **Input files**:
  - Many pairs of fastq files
  - Reference file
- **Output files**: aligned .sam files

Ben Bioinformatics

Needs to process 100s of genomic data files.

Use Case 1: Analyzing Multiple Files

- **Software**: bwa aligner
- **Executable**: Shell script with bwa commands
- **Arguments**: None (for now…)
- **Input files**:
  - Many pairs of fastq files
  - Reference file
- **Output files**: aligned .sam files

```
universe = container
container_image = bwa.sif

executable = bwa.sh
#arguments =

transfer_input_files = R1.fastq,
R2.fastq, ref.fastq, bwa.sif
#transfer_output_files =

error = test.err
output = test.out

queue 1
```

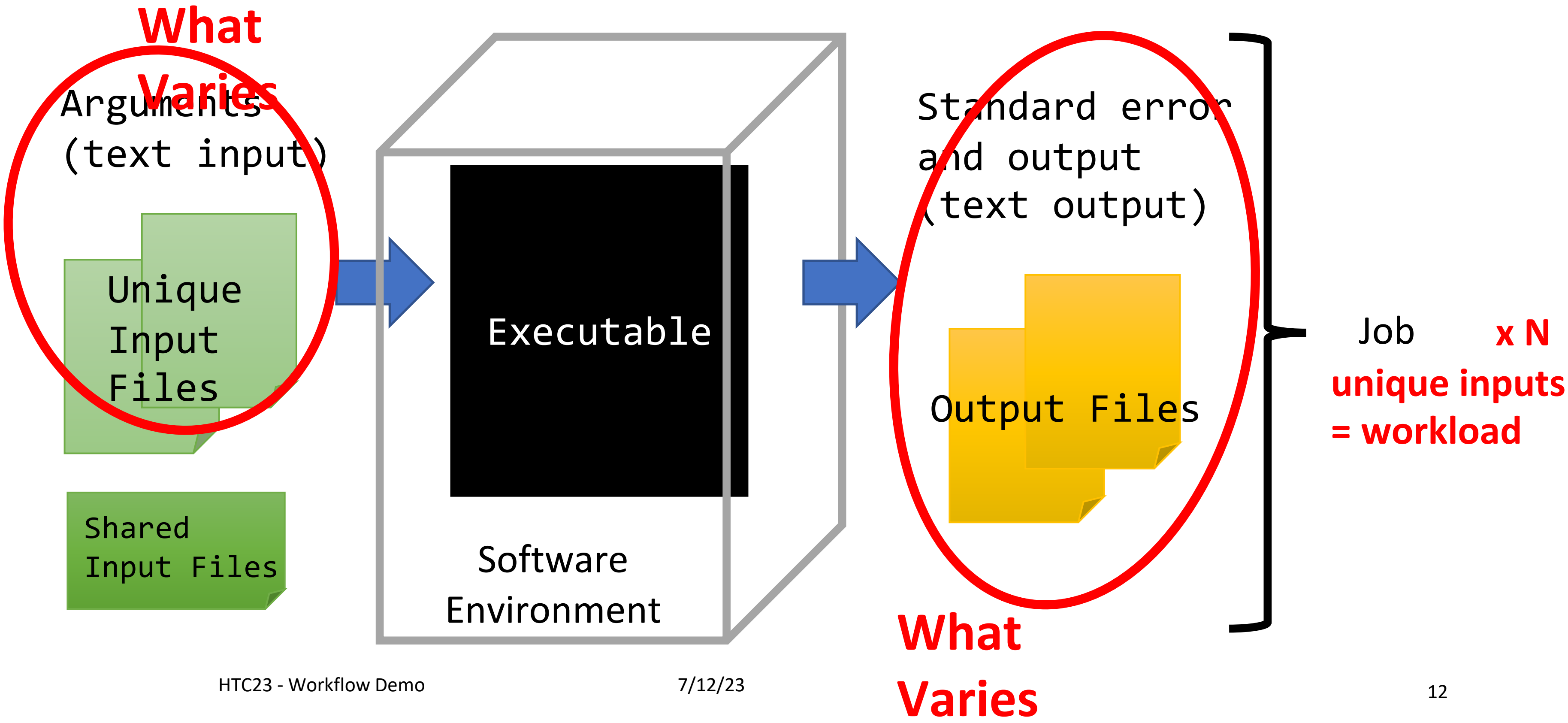
In Jupyter

In an opened terminal, run:

```
$ tutorial bwa
```

Then click on the downloaded folder (tutorial-bwa) and **open the “README.ipynb” file.**

Job Component Vocabulary - Expanded



Analyzing Multiple Files

```
executable = bwa.sh
#arguments =

transfer_input_files =
SRR1.R1.fastq, SRR1.R2.fastq,
ref.fastq, bwa.sif

transfer_output_remaps =
“SRR1.sam=results/SRR1.sam”

error = test.err
output = test.out

queue 1
```

```
executable = bwa.sh
arguments = $(sample)

transfer_input_files =
$(sample).R1.fastq,
$(sample).R2.fastq, ref.fastq,
bwa.sif
transfer_output_remaps =
“$(sample).sam=results/$(sample).sam”

error = test.$(sample).err
output = test .$(sample).out

queue sample from list.txt
```

In Jupyter

Continue working with the bwa tutorial.

Apply to Your Workflow

- Processing MRI or other imaging data
- Molecule/protein docking
- Simulations that are described by an input file
- Feature extraction
- …anything that has many unique input files, each representing a self-contained job producing unique output.

Building a Workload

Patterns for Scaling Out

- **“What is a job?”**
  - Define your unit of work and how many you need to run
  - Identify components (shared and unique/varied) of a single job
- **Generate Inputs**
  - Do you need to generate unique input files?
  - How about a list of inputs for your jobs?
- **Plan to summarize**
  - What steps, if any, are needed to combine results?

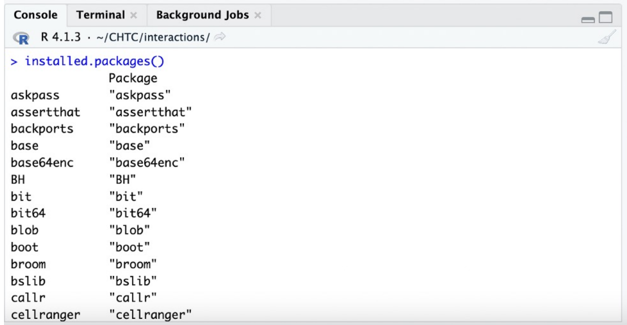
Patterns for Scaling Out

- **Write modular code**
  - Write one executable that 1) takes in unique inputs and 2) produces unique outputs.
- **Think about organization**
  - How do you want to arrange the components for your jobs?
- **Test, test, test**
  - Always test one job, then a small batch before doing a large run.
  - How much space is needed for job components?

Additional Considerations

- **Software environment**
  - Have to bring along a software environment
  - Containers – we provide a few, have directions how to build yourself
  - File-based – bring along binary files or zipped software directories
    - (Conda environments can be used this way)

APPTAINER

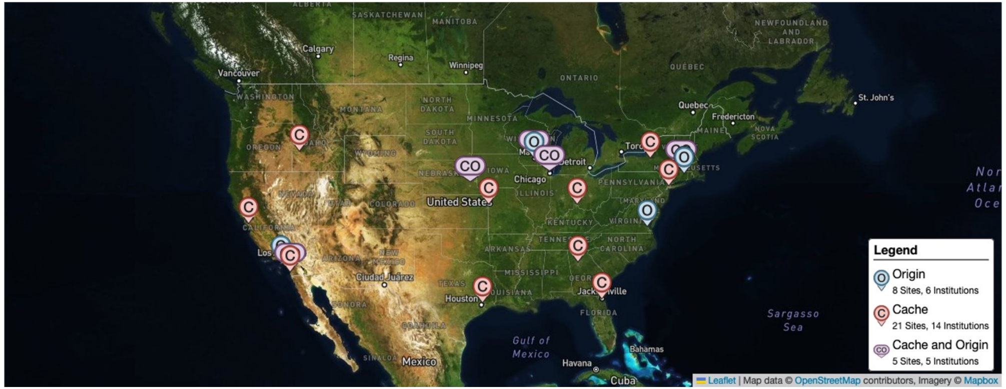


R 4.1.3 · ~/CHTC/interactions/

```
> installed.packages()
                Package
askpass         "askpass"
assertthat      "assertthat"
backports       "backports"
base            "base"
base64enc       "base64enc"
BH              "BH"
bit             "bit"
bit64           "bit64"
blob            "blob"
boot            "boot"
broom           "broom"
bslib           "bslib"
callr           "callr"
cellranger      "cellranger"
```

Additional Considerations

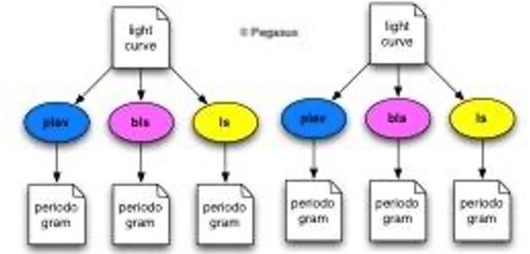
- **Data movement**
  - For input/output files between 1 – 20GB, need a scalable data staging tool
  - Open Science Data Federation
    - Network of data origins and caches to efficiently move data
  - Most OSPool Access Points have an associated data origin.

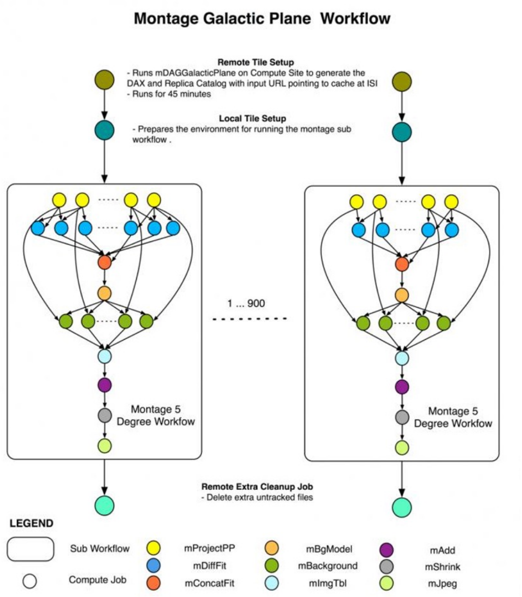


Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox

Additional Considerations

- **Multi-Step workflows**
  - DAGMan – comes with HTCondor
  - Pegasus - https://pegasus.isi.edu/





Montage Galactic Plane Workflow

Remote Tile Setup
- Runs mDAGGalacticPlane on Compute Site to generate the DAX and Replica Catalog with input URL pointing to cache at ISI
- Runs for 45 minutes

Local Tile Setup
- Prepares the environment for running the montage sub workflow .

Remote Extra Cleanup Job
- Delete extra untracked files

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2030508. 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.