HPC & Coding Basics for Economists

Ethan Hartley[1, 2]

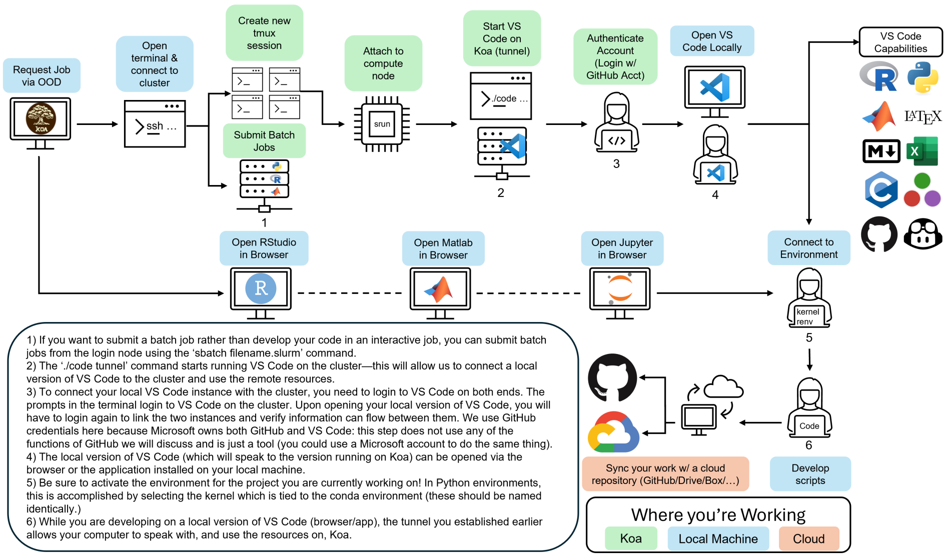
[1]University of Hawai‘i at Mānoa

[2]U.S. Department of Energy
(Oak Ridge National Laboratory)

Setting Up Koa: 4/7 & 4/8

- Provisional access to Koa will end on 4/9 (need confirmation)
- Email ehartley@hawaii.edu to setup a meeting to go through any material from the first two workshops.

Overview of Workshop 1



1) If you want to submit a batch job rather than develop your code in an interactive job, you can submit batch jobs from the login node using the 'sbatch filename.slurm' command.
2) The './code tunnel' command starts running VS Code on the cluster—this will allow us to connect a local version of VS Code to the cluster and use the remote resources.
3) To connect your local VS Code instance with the cluster, you need to login to VS Code on both ends. The prompts in the terminal login to VS Code on the cluster. Upon opening your local version of VS Code, you will have to login again to link the two instances and verify information can flow between them. We use GitHub credentials here because Microsoft owns both GitHub and VS Code: this step does not use any of the functions of GitHub we will discuss and is just a tool (you could use a Microsoft account to do the same thing).
4) The local version of VS Code (which will speak to the version running on Koa) can be opened via the browser or the application installed on your local machine.
5) Be sure to activate the environment for the project you are currently working on! In Python environments, this is accomplished by selecting the kernel which is tied to the conda environment (these should be named identically.)
6) While you are developing on a local version of VS Code (browser/app), the tunnel you established earlier allows your computer to speak with, and use the resources on, Koa.

Agenda: Data Transfers & Documentation

Data Types

- **Common Types**: Integer, Float, String, Boolean, Complex
- **Python**: Lists, Dictionaries, Sets
- **R**: Vectors, Data Frames, Factors
- **Optimized Types**: Use `int8, float32, category` for efficiency

Efficient Storage & Formatting

Storage Strategies:

- **Compressed Formats**: Zip, Gzip, and BZ2 compression may reduce size but can also slow read times.
- **Columnar vs. Row-based**:
    - **Row-based** (CSV, JSON, SQL): Good for transactional workloads where data is frequently inserted, updated, or deleted. These formats store records sequentially, making them suitable for fast row retrieval but less efficient for analytical queries on large datasets.
    - **Columnar** (Parquet, ORC): Best for analytical processing where specific columns are queried frequently. Columnar storage reduces disk I/O and enables better compression, leading to faster operations and efficient scanning.
- **Sparse Data**: Sparse representations like SciPy sparse matrices or Pandas SparseDataFrame store only non-zero values, reducing memory and computation time.

Choosing Data Formats

Format	**Type**	**Key Features**
CSV	Row-based	Simple, human-readable, large file sizes
JSON	Row-based	Flexible, hierarchical, human-readable
Parquet	Columnar	Compressed, optimized for large datasets
ORC	Columnar	Optimized for Hadoop, better compression than Parquet
HDF5	Hierarchical	Supports structured and unstructured data
NetCDF (.nc)	Multidimensional	Designed for array-based data like climate models
Feather	Columnar	Optimized for fast I/O with pandas/R
Avro	Row-based	Schema evolution support, binary storage
Excel (.xlsx, .xls)	Row-based	Widely used, supports formulas, multiple sheets
RData (.rda, .rds)	Binary	Efficient storage for R objects
Stata (.dta)	Row-based	Designed for statistical software, supports metadata

*It's important to choose the most efficient format to minimize unnecessary overhead. However, if data transfers and storage become limiting constraints, consider reaching out to ITS professionals for assistance.

Key Packages & Strategies for Storing Data

Python:

- `pandas`: General data manipulation and analysis.
- `polars`: DataFrame library for performance and parallelization.
- `dask`: Parallel computing library for scaling workflows across cores/nodes.

Key Packages in R:

- `dplyr`: Intuitive package for data manipulation.
- `data.table`: High-performance package for large datasets.
- `tidyr`: Package for tidying, reshaping, and cleaning data.
- `readr`: Fast package for reading and writing data.

In general, it's better to load multiple smaller files instead of very large files (> 10GB). For a guide on using various compression strategies on a Linux HPC, refer to Workshop_2 in the HPC-in-Econ-Basics repository.

Benefits of Using SQL Databases in HPC

- **Efficient Storage:** Structured storage for fast querying and retrieval of large datasets.
- **Scalability:** Handle millions of rows with ease, unlike traditional file storage.
- **Concurrent Access:** Ideal for multi-user environments on HPC systems.
- **Optimized Queries:** Indexing accelerates complex queries and boosts performance.
- **Data Integrity:** `ACID` properties ensure reliable data and backups.

SQL on Koa

SQLite (for individuals):

- Lightweight, serverless SQL database engine.
- Store all project data in a single database for personal or small-scale use.
- `sqlite3`: Built-in Python library for interacting with SQLite.

PostgreSQL (for multi-user environments):

- A more robust SQL database suitable for multi-user access.
- Ideal for centralizing datasets, allowing multiple users to access the same database concurrently.
- `psycopg2`: Python package for interacting with PostgreSQL.
- `SQLAlchemy`: SQL toolkit and ORM library for Python, supports both SQLite and PostgreSQL.

Examples: Using SQL on Koa

Intro to Data Transfers

Koa has a set of **Data Transfer Nodes (DTNs)** designed specifically for efficient data movement. You can connect to these nodes using: `ssh <username>@koa-dtn.its.hawaii.edu`

General Considerations for Data Transfers:

- **Speed vs. Reliability** – Large files may require tools that support resumption in case of failure.
- **Network Efficiency** – Parallel and compressed transfers can significantly reduce transfer time.
- **Security** – Always use encrypted methods such as `scp`, `rsync` over SSH, or `sftp`.
- **Automation** – Consider scripts for recurring transfers to streamline workflows.

Common Tools for Transferring Data

Using the right tools and strategies ensures **fast, reliable, and secure data transfers**.

- File Explorer/IDEs - Simple transfers for small files.
- `Globus` – High-performance, managed transfers for large datasets.
- `scp` – Secure, simple file transfer.
- `rsync` – Efficient for syncing directories and resuming transfers.
- `sftp` – Interactive file transfers.

Note: command line transfer & retrieval strategies can be used within tmux sessions.

*Koa is a centrally managed system and may not meet all data security requirements. While it is generally suitable for most research data, it is not unanimously appropriate for storing highly sensitive medical or financial information. Before transferring data to Koa, verify that it complies with any necessary security and confidentiality guidelines.

File Explorer

- In VS Code, RStudio, or JupyterLab, you can upload or download files by right-clicking on a file and selecting the appropriate option.
    - In VS Code, you can drag and drop files into the workspace to upload them.
    - This method is recommended for downloading figures, tables, and scripts—especially for files under 25MB and for low volume transfers.
- You can also use the File Explorer located under the ”Files” tab on Open OnDemand or within an Interactive Desktop job through Open OnDemand.

Globus

Globus is a robust data transfer and sharing platform designed for research and academic institutions. It provides secure, high-speed file transfers between endpoints, making it ideal for large datasets and HPC workflows. Key benefits include:

- Automated, fault-tolerant transfers with checkpointing and retries.
- Integration with institutional authentication for secure access.
- Support for both on-premise and cloud storage systems.
- Ability to share data with collaborators without manual file transfers.

▸ Globus on Koa

Data Transfers: Local ↔ Koa

Using `scp` (Secure Copy)

- Push a file from local to HPC: `scp myfile.txt username@koa-dtn.its.hawaii.edu:/remote/path/`
- Pull a file from HPC to local: `scp username@koa-dtn.its.hawaii.edu:/remote/path/myfile.txt .`
- Transfer directories (use `-r` for recursion): `scp -r myfolder username@koa-dtn.its.hawaii.edu:/remote/path/`

Using `sftp` (Secure FTP)

- Connect to HPC: sftp username@koa–dtn.its.hawaii.edu
- Transfer files:
  - Download: `get remote_file local_destination`
  - Upload: `put local_file remote_destination`
- Exit: exit

Using rsync (Efficient Synchronization)

Basic Commands

- Sync local to HPC:
  rsync –av myfolder username@koa–dtn.its.hawaii.edu:/remote/path/
- Sync HPC to local:
  rsync –av username@koa–dtn.its.hawaii.edu:/remote/path/myfolder .
- Resume interrupted transfers (`-P` for progress):
  rsync –avP largefile username@koa–dtn.its.hawaii.edu:/remote/path/

Best Practices

- Use `-P` with `rsync` to monitor progress.
- Use `scp -r` or `rsync -a` for directories.
- Test `rsync` with `--dry-run` before large syncs.

Data Transfers: Koa $\leftrightarrow$ Cloud

`Rclone` is a command-line tool for syncing and managing files across cloud storage providers like Google Drive, Dropbox, AWS S3, and OneDrive, making it ideal for transferring large datasets between Koa and the cloud.

- Use `rclone config` to set up remote storage.
- `rclone sync` can be used instead of `copy` for full synchronization.

Listing Files

- `rclone lsf uh_gdrive:`
- `rclone lsf uh_gdrive:Data/`

Transferring Files

- Copy a file to Google Drive: `rclone copy test uh_gdrive:test`
- Download a directory from Google Drive: `rclone copy uh_gdrive:Data Data`

▸ Example: Transferring Data Using Rclone

Downloading Data via the Command Line

Using `wget`:

- `wget` is a simple tool for downloading files over HTTP, HTTPS, and FTP.
- It supports recursive downloads, resuming interrupted downloads, and downloading entire websites.

Using `curl`:

- `curl` is a more versatile tool, supporting various protocols (HTTP, HTTPS, FTP, etc.), and is ideal for interacting with APIs.
- It supports downloading files as well as uploading, sending requests with custom headers, and more.

Key Differences:

- `wget` is simpler and automatically handles file downloading and resuming.
- `curl` is more flexible and suitable for complex interactions with URLs, such as posting data or handling APIs.

Examples of wget & curl Commands

Example Commands for `wget`:

- Download a file: `wget https://example.com/file.txt`
- Resume an interrupted download: `wget -c https://example.com/file.txt`
- Download files listed in a text file: `wget -i urls.txt`

Example Commands for `curl`:

- Download a file: `curl -O https://example.com/file.txt`
- Download and save to a specific file: `curl -o savedfile.txt https://example.com/file.txt`
- Download files listed in a text file: `curl -O $(cat urls.txt)`

When downloading lists of files, be conscious of request restrictions and overloading the website.

▸ Examples: NBER, Data.gov Archive, NASA

Data APIs

- APIs allow you to access and download data from remote servers or services programmatically.
    - They allow for the automatic retrieval of large datasets, eliminating the need to manually download each file or understand the backend of the database.
    - Always check if the resource provides an API for data access, and if the API supports bulk downloads.
    - Review the API documentation for limits on data access, authentication, and the format of data returned.
    - Make sure to follow any rate limits or usage restrictions to avoid overloading the service.
- Automate the process of retrieving regularly updated datasets.
- Save time by downloading data programmatically rather than manually.
- APIs often support filtering and querying data, allowing you to download only the necessary information.

Using Koa as a Conduit

When working with large datasets or files, Koa provides a seamless environment for managing data transfers. One useful tool is rclone, which allows us to easily download files to remote storage, such as Google Drive, directly from the source.

- `rclone copyurl [URL] uh_gdrive:/path/to/directory`

Breakdown:

- `[URL]`: The URL of the file you want to copy.
- `uh_gdrive`: The configured remote for your Google Drive.
- `/path/to/directory`: The directory within your Google Drive where the file should be placed.

This method allows for a quick and efficient transfer of files directly into your Google Drive without manual intervention. It is especially useful for batch processing and automated workflows in Koa.

Introduction to GitHub

GitHub is a cloud-based platform for version control and collaboration. Key Functionalities of GitHub include:

- **Version Control**: Track changes to your code over time.
- **Collaboration**: Work with others using branches and pull requests.
- **Repositories**: Store, manage, and share code.
- **Issue Tracking**: Report and track bugs or feature requests.

Example Repositories:

- Personal Repositories: Working Repository, Mini Results
- Replication Packages: Athey et al. (2022)
- Packages: statcanR, did
- Enterprise Level: Pytorch

Why should you care?

- **Searchability**: Easily find and reference past work.
  - You can search entire repositories for specific keywords instantaneously. **This is also a great use of Copilot!**
- **CYA**: GitHub's version control ensures your work is always backed up and recoverable.
  - Maintains a record of contributions and changes for collaborative work.
- **Issue Tracking**: Streamline problem-solving and project management.
- **Open-Source and Transparent Research**: Boosts credibility, particularly in interdisciplinary fields where GitHub is widely used.
- **Increased Impact**: Reduces the barriers to extending your research $\rightarrow$ hopefully netting you more citations.

Structure of GitHub

- **GitHub.com**: Navigating repositories, issues, and pull requests.
  - Creating and cloning repositories, Codespaces, and more advanced functions (merging and forking repos)
    - The hardest part about creating a repository is identifying the license you want to use.
- **GitHub Desktop**: A graphical user interface (GUI) for managing repositories.
  - Simplifies commit history visualization and branching.
  - Allows you to manage repositories as folders on your local machine.
- **GitHub CLI**: A command-line interface for managing repositories, issues, and pull requests from the terminal.
  - Provides automation capabilities for advanced users. However, most users prefer the GUI over the CLI for its ease of use.

How to Use in Practice

- **General Order of Operations**:
  - Clone or initialize a repository.
  - Make changes locally.
  - Commit your changes.
  - Push changes to GitHub.
  - Pull changes from GitHub to stay updated (relevant if you make changes through the browser).
- **Committing**: Save your changes locally with meaningful commit messages.
- **Pushing**: Upload your committed changes to the remote repository.
- **Pulling**: Download and merge changes from the remote repository.

*Note: you can upload files directly to GitHub repos via the browser.

Personal Portfolios

GitHub can be used to host your personal portfolio through platforms like Techfolios or Jekyll.

Techfolios:

- Documentation and guides can be found here.
- Clone template repositories at: Techfolios Templates.

Examples

- If you find style elements you like, you can visit their GitHub repositories to see how they implemented them:
  - `https://nick-ai.github.io/`
  - `https://aribunnell.github.io/`
  - `https://linneawolniewicz.github.io/`
  - `https://ethanhartley22.github.io/`
  - `https://peterjsadowski.github.io/`

Storing Large Datasets

Individual files on GitHub cannot exceed 100 MB. If you want to host your data and it is too large, or is unable to be split into smaller pieces, you can host open-source data freely through Hugging Face.

- Hugging Face is a platform for sharing and collaborating on machine learning models, datasets, and other AI resources.
  - Just because data isn't AI specific, doesn't mean it can't be stored here.
- It supports datasets in various formats and integrates easily with machine learning workflows.
- You can upload, access, and version control large datasets using Hugging Face's Dataset Hub.
  - Allows collaborators to load your data directly into their environment.
- You can upload data using the huggingface_hub package in Python or via the website.

Hugging Face Datasets Hub

Workshop 3: 4/15/25

- Workshop 1: Coding & HPC Basics
- Workshop 2: Data Transfers & Documentation
- Workshop 3: AI & ML in Economics
  - AI Fundamentals
    - Universal Principles
    - Implementation Strategies
  - Causal Machine Learning
  - Recent Advancements in AI
    - Potential applications in economics