Title	CWL in Practice: Adopting Common Workflow Language for existing research and academic computing clusters
Author	*Dan Leehr*, Alejandro Barrera, Hilmar Lapp
Affiliation	https://www.genome.duke.edu
Contact	dan.leehr@duke.edu
URL	https://github.com/Duke-GCB/GGR-cwl
License	MIT

Assembling individual bioinformatics software packages to run together as a pipeline is often done by scripting them together in a language like Bash, Python, or Perl. This approach, while simple to build and understand, often yields pipelines that are not generalizable across data sets or computing environments. Further, tracking or archiving versions of packages requires nontrivial effort, as does engineering for modularly. These properties hinder reproducibility of the process, as reconstructing the pipeline can be cumbersome or impossible without knowledge of softare versions or access to the original enivornment.

In contrast to the above, the Common Workflow Language provides a blueprint for architecting workflows. With open source implementations and a comprehensive specification, it lowers the barrier to building reusable components and reproducible pipelines for bioinformatics. These pipelines can be run on any implementation supporting the language, and thus are a desirable product for reproducible research. Additionally, CWL encourages Docker images, bundling tools and their dependencies completely. The images thus allow compute environments to be perfectly archived and published.

At Duke Center for Genomic and Computational Biology (GCB), we're using CWL in support of the Genomics of Gene Regulation project. The project aims to develop new pipelines for processing data from ChIP-seq, RNA-seq, and DNase-seq experiments. By building workflows that conform to the CWL standard, we'll be able to distribute reproducible pipelines with publications and data sets. During the research and development of these CWL pipelines, we've met two discrete challenges: Translating existing logic to CWL and integrating with our high-performance computing resources. In this presentation, we will discuss our approaches to addressing these challenges.

We'll cover the details of translating existing control flow and imperative scheduling logic to CWL's declarative language. This includes pragmatic approaches to reusing components for manageable workflows.

We'll also discuss the practical aspects of adopting CWL in an academic HPC environment. Our center operates a 40-node HPC cluster running Slurm for computational genomic workloads. As it is a shared compute environment, it is not suitable to run Docker or virtual machines favored by CWL. However, publishing Docker images makes it easier for consumers to use our workflows. We build Docker images and load modules with helmod, both providing the necessary environment for CWL tools to run. Finally, we'll address how we're extending toil, a workflow engine that supports CWL with Slurm support, allowing us to scale up CWL workflows to our available compute resources.