WDL: The Workflow Description Language

Jeff Gentry
Kate Noblett
Chris Llanwarne
Geraldine Van der Auwera
Workshop Agenda

- What’s WDL (15 min)
- Hands on introduction (1 hr)
- Break (5 min)
- Exploring the ecosystem (40 min)
- Hands on exercises (30 min)
The backdrop: data generation set to explode

Quarterly output (in TBases) of the Genomics Platform

Story begins here
Plenty of workflow solutions to go around

So of course we decided to create a new one.

Randall Munroe, XKCD
https://www.xkcd.com/927/
We needed a workflow language that ...

• Humans can read/write
  – Tool developers and biomedical scientists at large
  – “Programming language-lite”
• Is portable
  – Decoupled from the engine
  – Not tied to a specific programming environment
  – Independent of underlying execution platform
TL;DR: We are handing over governance of WDL to an open community-based organization called OpenWDL that will ensure WDL development is driven by and for the research community.

When WDL was created a few years ago, our mission was ostensibly to produce a pipelining system for the Broad's internal needs. But we knew at the time it would have to be much more than just a Broad-specific tool, because we had already decided we would share our analysis pipelines as widely as possible. The most immediate consequence was that whatever pipelining solution we adopted needed to be portable across not only our method development and production environments, but also across a variety of systems used in the research community. The other was that the scripting language we used needed to be as user-friendly as possible, to be at the very least readable by e.g. biomedical scientists without formal computational training (which happens to be my own background, so I have an especially soft spot for that constituency). Those were huge factors that led to our decision to create a new engine (Cromwell) and domain language (WDL itself, obviously :)) rather than adopt an existing system. It also motivated us to provide as much help (via the forum) and documentation as we could manage to anyone who expressed interest, whether they were Broadies or not.

Fast-forward to the present day, where we’re starting to see individuals and organizations throughout the world adopt WDL (sometimes independently of Cromwell) for their own pipelining needs. This is a incredibly gratifying but also rather daunting, because that means we now bear some degree of responsibility in ensuring that the people who adopt WDL are empowered to be successful in their pipelining endeavors.

This realization kicked off a bit of a chain reaction. Figuring we were going to need help to understand what are the needs of that community, let alone satisfy them, we reached out to external people we know and trust who have been using WDL fairly intensively. We thought perhaps we could assemble a sampling of these power users to act as an advisory group, maybe a focus panel. We had some very illuminating conversations as a result -- in the sense that they told us not what we wanted to hear, but what we needed to hear; that our approach was all wrong. They told us that what would most benefit the WDL user community is not for us to make an effort to take directions, but to hand over the keys and let the community drive.

So after we put our hairbrush back down from our hairline, we did some soul searching. Then we talked to the Broad's legal...
There’s an elephant in the room ...
Workflow description Language (WDL)

```wdl
workflow myWorkflowName {
  File my_ref
  File my_input
  String name

  call task_A {
    input: ref= my_ref, in= my_input, id= name
  }
  call task_B {
    input: ref= my_ref, in= task_A.out
  }
}

task task_A {
  File ref
  File in
  String id

  command {
    do_stuff -R ${ref} -I ${in} -O ${id}.ext
  }
  runtime {
    docker: "my_project/do_stuff:1.2.0"
  }
  output {
    File out= "${id}.ext"
  }
}

task task_B {
  ...
}
```
WDL runtime parameters

task echoHelloWorld {
    command {
        echo 'Hello, World!' 
    }
    runtime {
        docker: "phusion/baseimage"
        disks: "local-disk 10 HDD"
        memory: "1 GB"
        preemptible: 3
    }
}

workflow printHelloAndGoodbye {
    call echoHelloWorld
}
Basic WDL plumbing options

### LINEAR CHAINING

```
call stepA
call stepB { input: in=stepA.out }
call stepC { input: in=stepB.out }
```

### MULTI-IN/OUT

```
call stepC { input :
  in1=stepB.out1,
in2=stepB.out2 }
```

### SCATTER-GATHER

```
Array[File] inputFiles
scatter(oneFile in inputFiles) {
  call stepA { input: in=oneFile }
}
call stepB { input: files=stepA.out }
```
There’s also a pig in the room ...
Two main ways to run Cromwell

**One-off**

- Simple self-contained command

  ```
  java -jar cromwell.jar \
  run hello.wdl \
  --inputs hello_inputs.json
  ```

- Appropriate for independent analysts

**Server mode**

- API endpoints
- More scalable
- Some devops needs
- Appropriate for production environments
- Call-caching! (aka “ka-ching”)
Steps to running a WDL in standalone mode

- Validate syntax
  ```java
  java -jar womtool.jar validate hello.wdl
  ```

- Generate inputs JSON
  ```java
  java -jar womtool.jar inputs hello.wdl > hello_inputs.json
  ```

- Run
  ```java
  java -jar cromwell.jar run hello.wdl --inputs hello_inputs.json
  ```
Ok, let’s get our hands dirty ...