June 15th Workshop Materials

Instructions for Hands-on Exercises

Mini Mutation Calling Exercise

Workspace: *broad-firecloud-workshops/ToolDevWorkshop_MiniMutationCalling_*
Methods: *broadinstitute/MiniMutationCalling* (ContEst, MuTect, Oncotator)
Method Configs: *mhanna/MiniMutationCalling_Cfg*
Data: 2 pairs of cell lines; a whole exome pair, and a “tiny” 100-gene pair of BAMs
Entities: participant, sample, pair
Results: Nozzle Report
Hands-on Steps:

This Mini Mutation Calling Tutorial includes a subset of tools from our complete Broad Mutation Calling Best Practice Workflow. It contains ContEst, MuTect, and Oncotator.

For this exercise, we will clone this workspace which is prepopulated with data and a method configuration. Then we will launch the MiniMutationCalling workflow analysis on 2 pairs of cell line BAMs, tiny BAMs containing only 100 genes and whole exome BAMs.

When run on the 100-gene BAMs, the expected runtime is roughly 30 minutes. When run on whole exome BAMs, the expected runtime is roughly 3 hours.

Steps

To run the Mini Mutation Calling Tutorial,

1. Navigate to the *broad-firecloud-workshops/ToolDevWorkshop_MiniMutationCalling_* workspace. You can copy this name and paste it into the search bar.
2. Clone the *broad-firecloud-workshops/ToolDevWorkshop_MiniMutationCalling_* workspace under the Google Project, *broad-firecloud-tutorials* and give it a unique name, e.g., *broad-firecloud-workshops/ToolDevWorkshop_MiniMutationCalling_<user name>*.
3. In your cloned workspace, navigate to the Data tab.
4. Upload TSV files in the correct order. First, be sure to download and unzip the *Workshop_Materials* folder. You will find the TSV files in a subfolder called *MiniMutCallingExercise*. In the Data tab, click **Import Data → Import from file → Choose file**. You should first upload from *MiniMutCallingExercise 1_participant.txt*, then *2_sample.txt*, then *3_pair.txt*. Review the Data tab to confirm that participant, sample, and pair data appears.
5. Navigate to the Method Configurations tab.
6. Select the *MiniMutationCalling_Cfg* Method Config and click **Launch Analysis**.

7. In the Launch Analysis window, toggle to pair and select *HCC1143_WE_pair*. This is a pair (tumor and normal) of whole exome BAMs on which to run this analysis. Click **Launch**.
8. Return to the Method Configurations tab and select the *MiniMutationCalling_Cfg* again. Click **Launch Analysis** and this time select *HCC1954_100_gene_pair.* This is a pair of tiny (100 gene) tumor/normal BAMs. Click **Launch**.
9. Check the Monitor tab in about 30 minutes to see if your analysis run on *HCC1954_100_gene_pair* completed. The analysis run for *HCC1143_WE_pair* finish in about 3 hours.
10. When the Monitor tab displays Done, click on the Data tab and, using the filtering widget on the left-hand side of the page, filter down to *Entity Name* and *nozzle_report*. Open the Nozzle Report and review the results of the run.

Tool Developer Exercises

0. Preliminaries

0.1 docker: `Cannot connect to Docker daemon`

If, when issuing a docker run command on your laptop console, you receive the message:

```
docker: Cannot connect to the Docker daemon. Is the docker daemon running
on this host?.
```

The following shell commands should address the problem:

```
$ docker-machine restart default
$ eval $(docker-machine env default)
```

0.2 What directory to run the exercises in

You should unzip the Workshop_Materials.zip archive if you have not already done so. You can run these exercises in any directories of your choosing. You will need write-access to the directory in which you run the exercises.

The instructions below assume that Workshop_Materials.zip file has been unzipped in the user's (birger) home directory, and all exercises are run in ~/test.

1. Hello World

In this exercise we work with a previously dockerized Hello World application written in Python. The Hello World Python script has been included with the zip archive (Workshop_Materials.zip > HelloWorld folder) downloaded from the FireCloud Forum. The following instructions assume the archive has been unzipped in your home directory and is running from your home directory. You may choose to run the exercise in any directory of your choosing, provided you have write access to that directory. The dockerized version of the application (a docker image) has been written the repository cbirger/hello-world on docker hub, with the tag “5.0”.

1.1 Test dockerized helloworld application in local docker container

1. Launch the Docker QuickStart Terminal
2. Pull the docker image cbirger/hello-world:5.0 from Docker Hub into your local image repository.

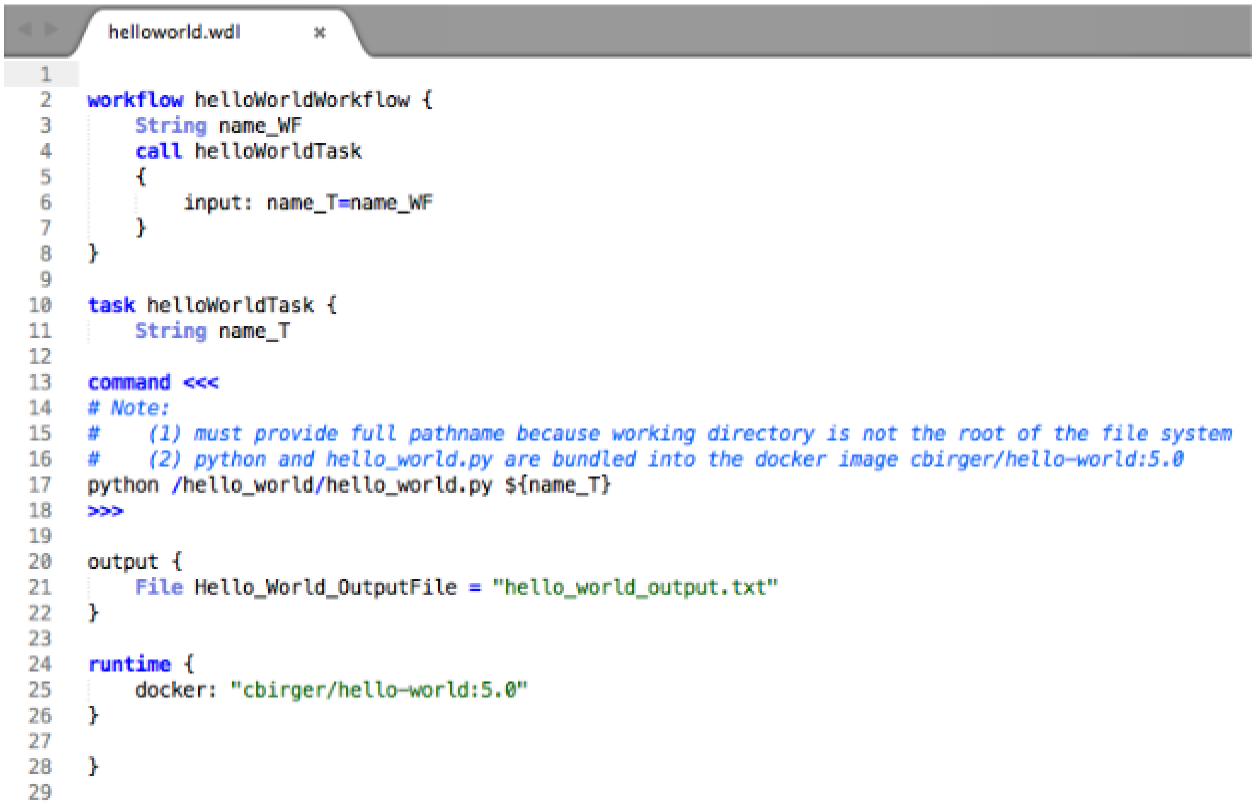
```
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \______ o           __/
             \    \         __/
              \____\_______/


docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

WMDAF-281:~ birger$ docker pull cbirger/hello-world:5.0
5.0: Pulling from cbirger/hello-world

51f5c6a04d83: Already exists
a3ed95caeb02: Pull complete
7004cfc6e122: Already exists
5f37c8a7cfbd: Already exists
8ad7684cace4: Pull complete
22c0978ecede: Pull complete
b1d402426f39: Pull complete
e7d853e7d62d: Pull complete
6abc0b6e5732: Pull complete
0d96eba6631d: Pull complete
ed7521fa4c17: Pull complete
4deeb48f16f4: Pull complete
Digest: sha256:52a4fcaf17a86248492858798926e48ba89e74725d6b6d32a2e1af1bebe8d0fb
Status: Downloaded newer image for cbirger/hello-world:5.0
WMDAF-281:~ birger$
```

3. Run helloworld application in docker container

```
WMDAF-281:test birger$ docker run cbirger/hello-world:5.0 python \
~/Workshop_Materials/hello_world/hello_world.py
Hello World!
WMDAF-281:test birger$ docker run cbirger/hello-world:5.0 python \
~/Workshop_Materials/hello_world/hello_world.py "Workshop"
Hello Workshop!
WMDAF-281:test birger$
```

1.2 Create single-task WDL workflow that calls helloworld and test it in locally running

We have already provided you with this WDL file (helloworld.wdl) in the zip archive mailed to workshop attendees.



```

workflow helloWorldWorkflow {
    String name_WF
    call helloWorldTask
    {
        input: name_T=name_WF
    }
}

task helloWorldTask {
    String name_T

command <<<
# Note:
#     (1) must provide full pathname because working directory is not the root of the file system
#     (2) python and hello_world.py are bundled into the docker image cbirger/hello-world:5.0
python /hello_world/hello_world.py ${name_T}
>>>

output {
    File Hello_World_OutputFile = "hello_world_output.txt"
}

runtime {
    docker: "cbirger/hello-world:5.0"
}

}

```

1.3 Test helloworld WDL running Cromwell locally

1. Use wdltool to validate syntax; blank response indicates the WDL is correct

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/wdltool-0.4.jar validate \
~/Workshop_Materials/HelloWorld/helloworld.wdl


WMDAF-281:test birger$
```

2. Create json template for workflow inputs

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/wdltool-0.4.jar inputs \
~/Workshop_Materials/HelloWorld/helloworld.wdl
{
  "helloWorldWorkflow.name_WF": "String"
}
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/wdltool-0.4.jar inputs \
~/Workshop_Materials/HelloWorld/helloworld.wdl > helloworld_input.json
WMDAF-281:~ birger$
```

Using your text editor, replace “String” above with the name of whom you want to greet; e.g., `"helloWorldWorkflow.name_WF": "Workshop"`

3. Run WDL with local Cromwell

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/cromwell-0.19.jar \
run ~/Workshop_Materials/HelloWorld/helloworld.wdl helloworld_input.json
[2016-06-12 14:04:13,432] [info] Slf4jLogger started
[2016-06-12 14:04:13,470] [info] RUN sub-command
[2016-06-12 14:04:13,471] [info]   WDL file: helloworld.wdl
[2016-06-12 14:04:13,472] [info]   Inputs: helloworld_input.json
...
...
[2016-06-12 14:04:22,826] [info] WorkflowActor [21ac8619]: transitioning from Running to
Succeeded.
{
  "helloWorldWorkflow.helloWorldTask.Hello_World_OutputFile":
"/Users/birger/Workshop/HelloWorld/cromwell-executions/helloWorldWorkflow/21ac8619-b0be-4127
-b16c-358c8d7cc0b7/call-helloWorldTask/hello_world_output.txt"
}
[2016-06-12 14:04:22,863] [info] SingleWorkflowRunnerActor workflow finished with status
'Succeeded'.
WMDAF-281:test birger$
```

4. Look at output

```
WMDAF-281:test birger$ ls
cromwell-executions   cromwell-workflow-logs       helloworld_input.json
WMDAF-281:test birger$ cd cromwell-executions/
WMDAF-281:cromwell-executions birger$ ls
helloWorldWorkflow
WMDAF-281:cromwell-executions birger$ cd helloWorldWorkflow/
WMDAF-281:helloWorldWorkflow birger$ ls
d818a24d-195f-4697-9123-2d645d6ff940
WMDAF-281:helloWorldWorkflow birger$ cd d818a24d-195f-4697-9123-2d645d6ff940/
WMDAF-281:d818a24d-195f-4697-9123-2d645d6ff940 birger$ ls
call-helloWorldTask
WMDAF-281:d818a24d-195f-4697-9123-2d645d6ff940 birger$ cd call-helloWorldTask/
WMDAF-281:call-helloWorldTask birger$ ls
hello_world_output.txt       rc                    script                stderr
       stdout
WMDAF-281:call-helloWorldTask birger$ cat hello_world_output.txt
Hello Workshop!
WMDAF-281:call-helloWorldTask birger$
```

1.4 Use firecloud CLI to upload WDL to FireCloud Method Repository

We do this from a docker container; it is easier to run the official broadinstitute/firecloud-cli docker image in a docker container than build and install the firecloud CLI to run natively on your machine.

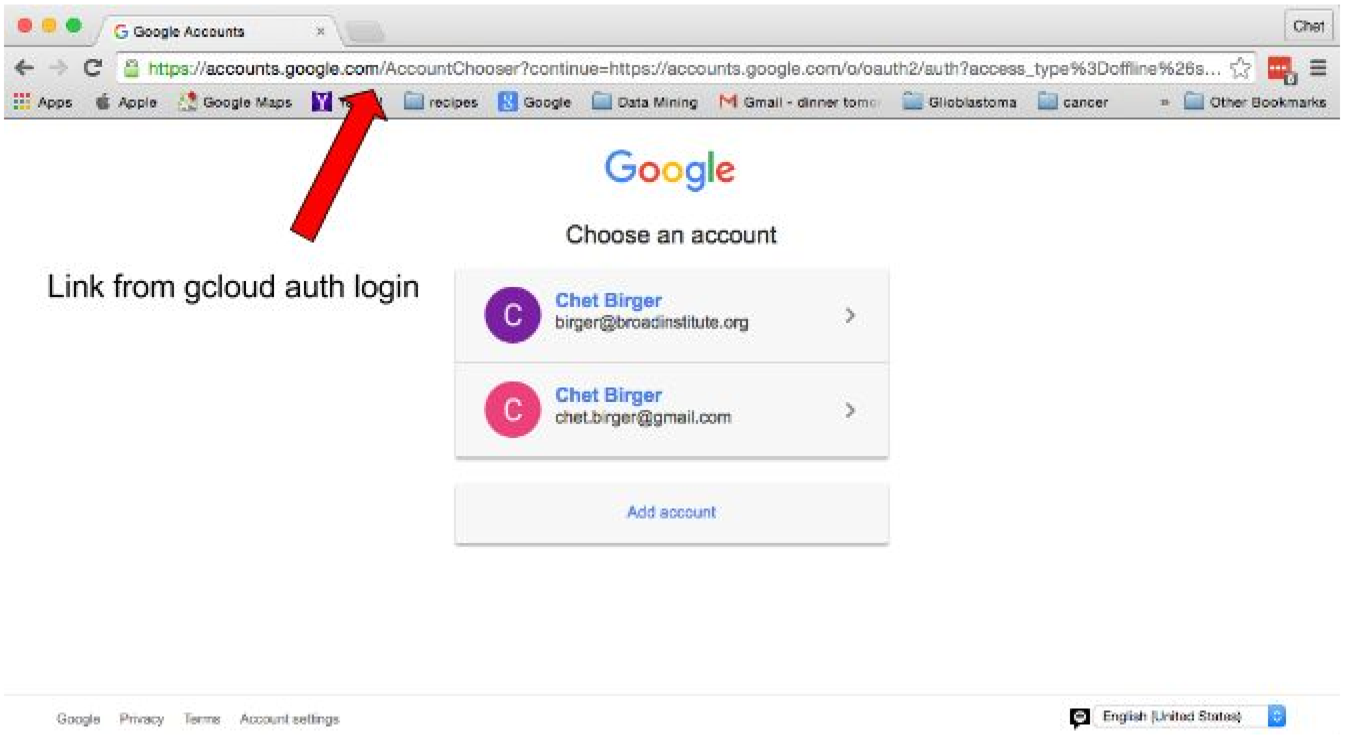
1. First make sure your docker engine is running locally (on a Mac OS X system, that means running the Docker quickstart terminal.)
2. Then cd to the directory where your wdl file resides.
3. From that directory launch an interactive bash shell in docker container loaded with the broadinstitute/firecloud-cli docker image

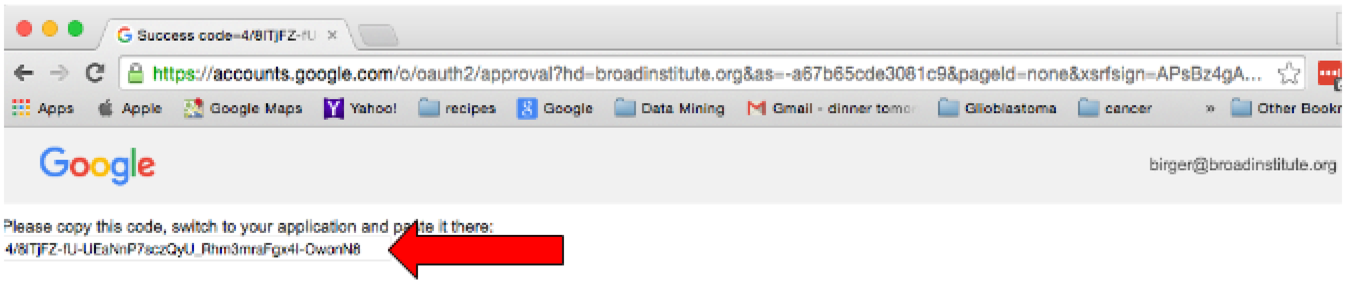
```
WMDAF-281:HelloWorld birger$ docker run --rm -it -v "$HOME"/.config:/.config -v \
"$PWD":/working broadinstitute/firecloud-cli bash
root@0a45e7c10a4e:/working#
```

- Note the mounting of your current working directory to /working in the container `-v "$PWD":/working`
- Note the mounting of the .config directory to the container's .config directory. `-v "$HOME"/.config:/.config` This will allow us to make user credentials available across multiple containers.

4. Within the running docker container run gcloud auth loginoror

```
root@0a45e7c10a4e:/working# gcloud auth login
Go to the following link in your browser:


https://accounts.google.com/o/oauth2/auth?redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0
%3Aoob&prompt=select_account&response_type=code&client_id=32555940559.apps.googleuse
rcontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A
%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2F
auth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute&access_type
=offline


Enter verification code:
```


Google Accounts
https://accounts.google.com/AccountChooser?continue=https://accounts.google.com/o/oauth2/auth?access_type%3Doffline%26s...
Link from gcloud auth login
Google
Choose an account
Chet Birger
birger@broadinstitute.org
Chet Birger
chet.birger@gmail.com
Add account
Google Privacy Terms Account settings
English (United States)


Success code=4/8ITjFZ-fU
https://accounts.google.com/o/oauth2/approval?hd=broadinstitute.org&as=-a67b65cde3081c9&pageId=none&xsrfsign=APsBz4gA...
Apps
Apple
Google Maps
Yahoo!
recipes
Google
Data Mining
Gmail - dinner tomo
Glioblastoma
cancer
Other Bookr
birger@broadinstitute.org

Please copy this code, switch to your application and paste it there:
4/8ITjFZ-fU-UEaNnP7aczQyU_Rhm3mraFgx4l-OwonN8

Copy and paste the verification code from your browser login session into the docker container session.

```
Enter verification code: 4/PoRnDE3Cd777A7gZL_xnPQDnHz6540im1ya7_JyNEoQ
Saved Application Default Credentials.

You are now logged in as [birger@broadinstitute.org].
Your current project is [None].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
root@0a45e7c10a4e:/working#
```

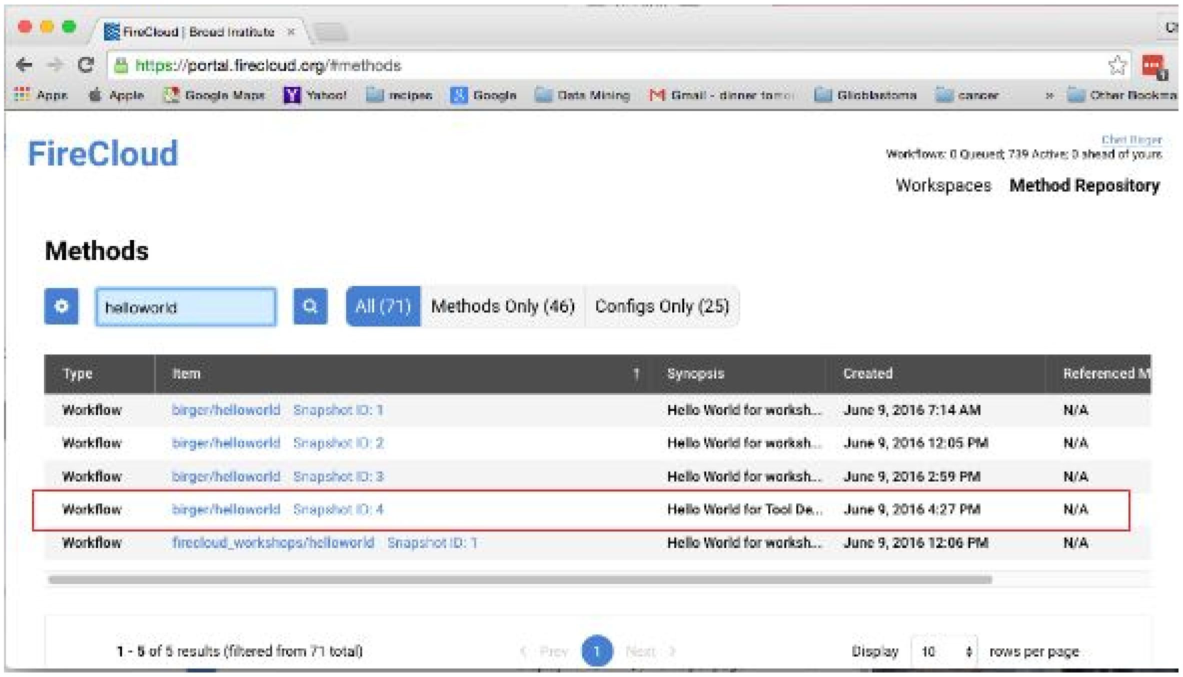
Now any firecloud commands run in the container will have access to credentials needed to authenticate to firecloud. (In addition, because the .config directory in the container binds back to .config on your local machine, new docker containers with the same binding can use those credentials.)

5. Within the same container run the firecloud cli command to upload a copy of your WDL to FireCloud's method repository:

Below is the docker command to upload a copy of your WDL to FireCloud’s method repository. Note that the -s option specifies the Method Repository namespace in which to publish the WDL. You will want to create your own namespace (replace “birger” with your own name).

```
root@0a45e7c10a4e:/working# firecloud -m push -s birger -n helloworld \
-t Workflow -y "Hello World for Tool Developers Workshop" helloworld.wdl
Succesfully pushed. Reponse:
{
  "name": "helloworld",
  "createDate": "2016-06-09T20:27:53Z",
  "payload": "workflow helloWorldWorkflow {\n\tString name_WF\n\tcall
helloWorldTask\n\t{\n\t\tinput: name_T=name_WF\n\t}\n}\n\ntask helloWorldTask
{\n\tString name_T\n\ncommand <<<\n# Note: \n#    (1) must provide full pathname
because working directory is not the root of the file system\n#    (2) python and
hello_world.py are bundled into the docker image cbirger/hello-world:5.0\npython
/hello_world/hello_world.py ${name_T}\n>>>\n\noutput {\n\tFile
Hello_World_OutputFile = \"hello_world_output.txt\"\n}\n\nruntime {\n\tdocker:
\"cbirger/hello-world:5.0\"\n}\n\n}",
  "url":
"http://agora.dsde-prod.broadinstitute.org/api/v1/methods/birger/helloworld/4",
  "documentation": "",
  "synopsis": "Hello World for Tool Developers Workshop",
  "entityType": "Workflow",
  "snapshotId": 4,
  "namespace": "birger"
}
root@0a45e7c10a4e:/working#
```

6. Go to Method Repository to see your uploaded method



Workflows: 0 Queued; 739 Active; 0 ahead of yours

Methods

helloworld

All (71) Methods Only (46) Configs Only (25)

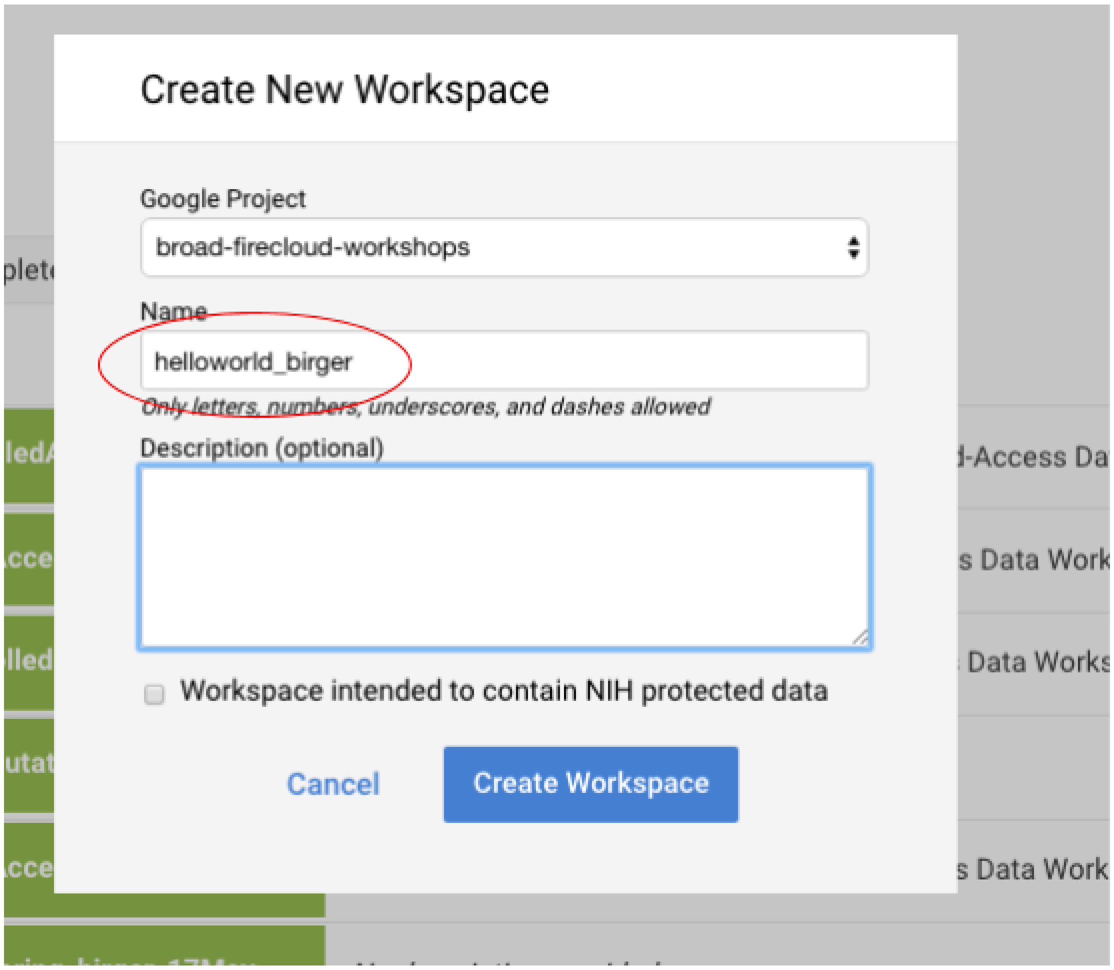
Type	Item	Synopsis	Created	Referenced M
Workflow	birger/helloworld Snapshot ID: 1	Hello World for worksh...	June 9, 2016 7:14 AM	N/A
Workflow	birger/helloworld Snapshot ID: 2	Hello World for worksh...	June 9, 2016 12:05 PM	N/A
Workflow	birger/helloworld Snapshot ID: 3	Hello World for worksh...	June 9, 2016 2:59 PM	N/A
Workflow	birger/helloworld Snapshot ID: 4	Hello World for Tool De...	June 9, 2016 4:27 PM	N/A
Workflow	firecloud_workshops/helloworld Snapshot ID: 1	Hello World for worksh...	June 9, 2016 12:06 PM	N/A

1 - 5 of 5 results (filtered from 71 total) Prev 1 Next Display 10 rows per page

1.5 Run the tool within FireCloud

You will now run the helloworld method on data you upload to a FireCloud workspace. You will create a new workspace, add participant entities to that workspace, create a method configuration in your workspace that maps the inputs and outputs of your helloworld method to participant entity attributes, and run the helloworld method on those workspace entities.

1. Create a new workspace by clicking on Create New Workspace... on the FireCloud portal's main page. The workspace will be charged to the FireCloud Billing Project `broad-firecloud-workshops`. Workspace names within that project must be unique. Name your workspace `helloworld_<username>`.



Create New Workspace

Google Project: broad-firecloud-workshops

Name: helloworld_birger

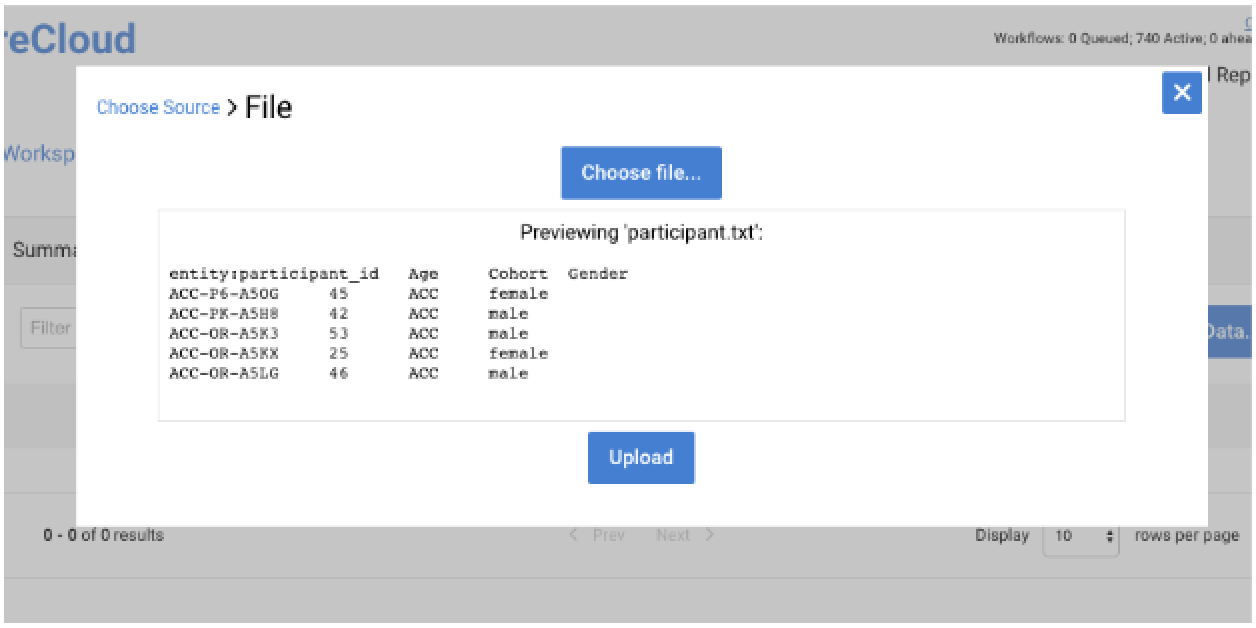
Only letters, numbers, underscores, and dashes allowed

Description (optional):

☐ Workspace intended to contain NIH protected data

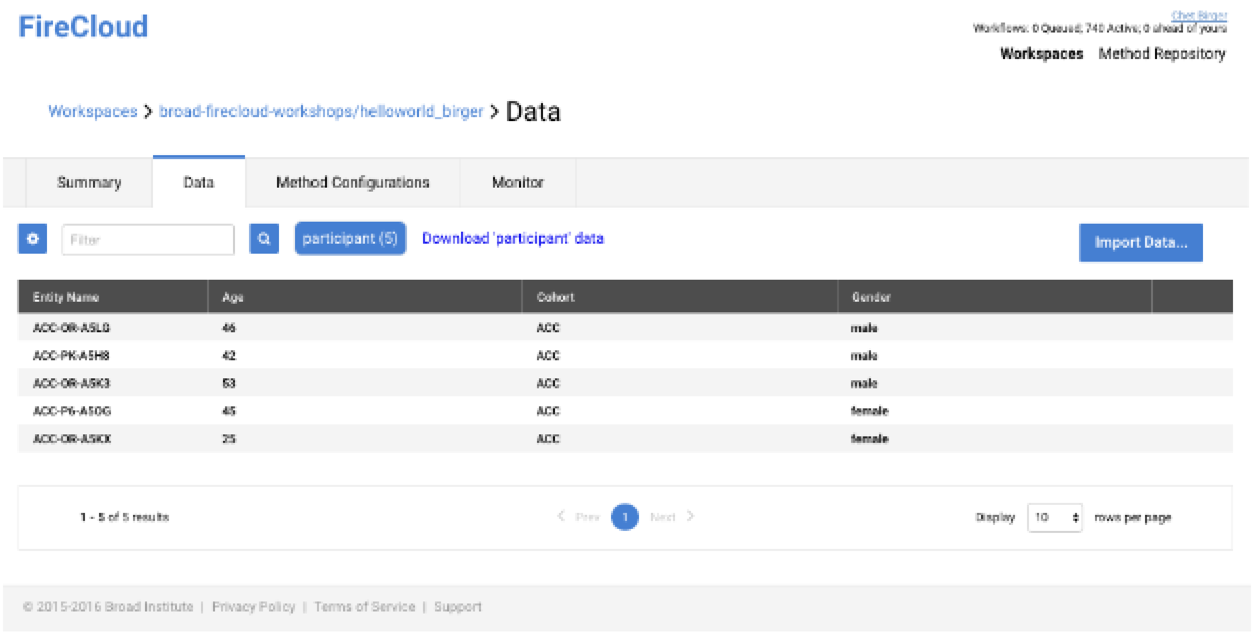
2. Upload participant entities to the new workspace

- Go to the data tab, select Import Data... → Import from file… → Choose file...
- Choose the participants.txt TSV file provided in the zip archive emailed to all workshop attendees.



Previewing 'participant.txt':

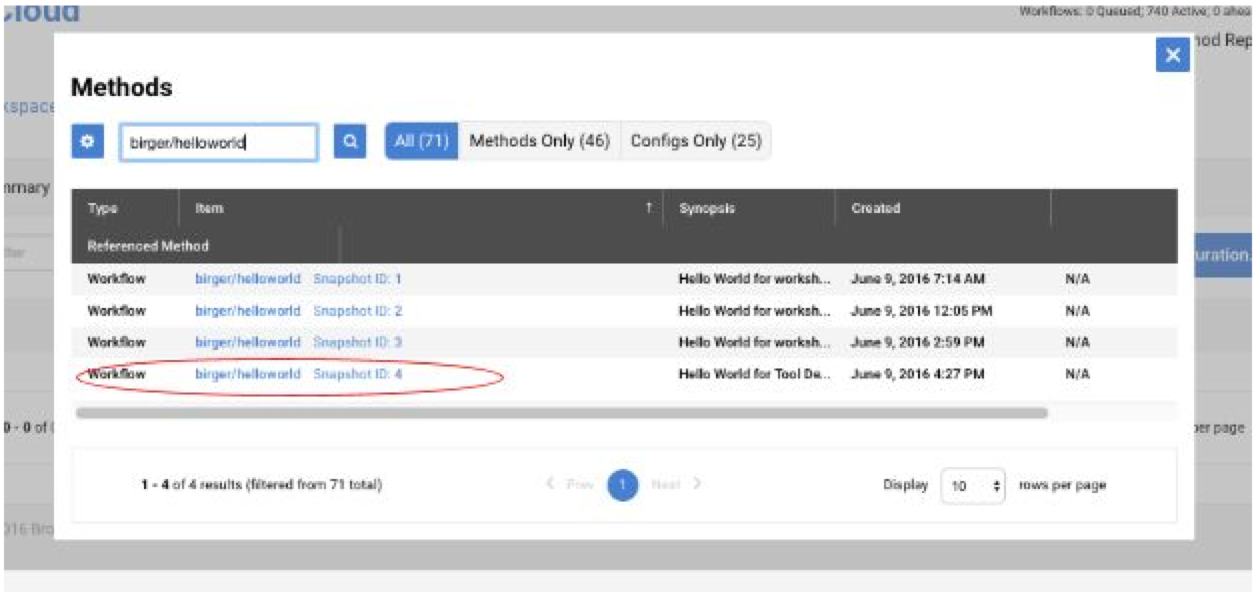
```
entity:participant_id	Age	Cohort	Gender
ACC-P6-A5OG	45	ACC	female
ACC-PK-A5H8	42	ACC	male
ACC-OR-A5K3	53	ACC	male
ACC-OR-A5KX	25	ACC	female
ACC-OR-A5LG	46	ACC	male
```


FireCloud
Workflows: 0 Queued; 740 Active; 0 ahead of yours
Workspaces
Method Repository
Workspaces > broad-firecloud-workshops/helloworld_birger > Data
Summary
Data
Method Configurations
Monitor
Filter
participant (5)
Download 'participant' data
Import Data...
1 - 5 of 5 results
Prev
1
Next
Display 10 rows per page
© 2015-2016 Broad Institute | Privacy Policy | Terms of Service | Support

Entity Name	Age	Cohort	Gender
ACC-OR-A5LG	46	ACC	male
ACC-PK-A5H8	42	ACC	male
ACC-OR-A5K3	53	ACC	male
ACC-P6-A5OG	45	ACC	female
ACC-OR-A5KX	25	ACC	female

3. Create a Method Configuration

- Go to the Method Configurations tab and click Import Configuration...
- Filter on <your namespace>/helloworld and select the helloworld workflow you previously uploaded to the method repository.



Methods

birger/helloworld

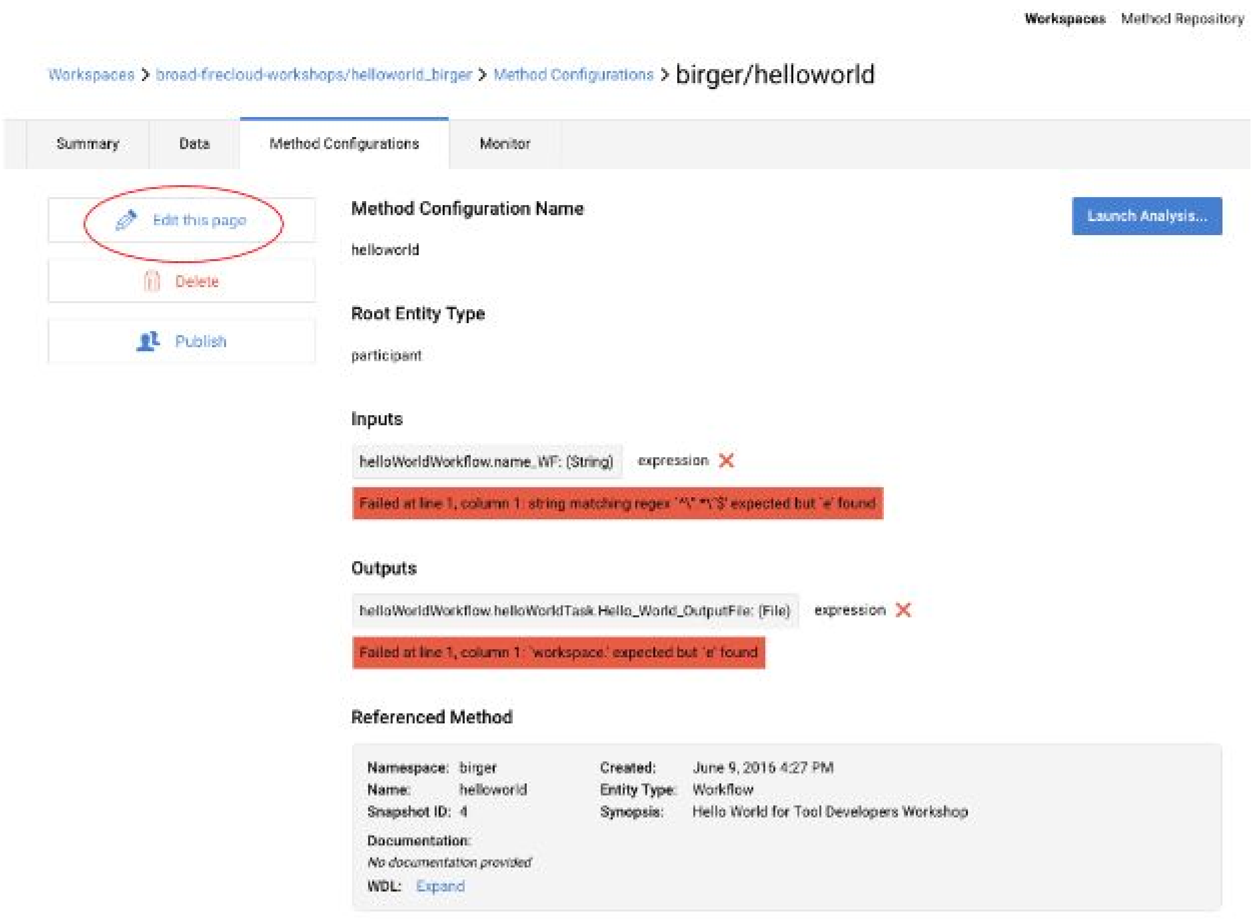
All (71) | Methods Only (46) | Configs Only (25)

Type	Item	Synopsis	Created	Referenced Method
Workflow	birger/helloworld Snapshot ID: 1	Hello World for worksh...	June 9, 2016 7:14 AM	N/A
Workflow	birger/helloworld Snapshot ID: 2	Hello World for worksh...	June 9, 2016 12:05 PM	N/A
Workflow	birger/helloworld Snapshot ID: 3	Hello World for worksh...	June 9, 2016 2:59 PM	N/A
Workflow	birger/helloworld Snapshot ID: 4	Hello World for Tool De...	June 9, 2016 4:27 PM	N/A

1 - 4 of 4 results (filtered from 71 total)

Display 10 rows per page

- Select the desired Workflow, and then click Import



Method Configuration Name

helloworld

Root Entity Type

participant

Inputs

helloWorldWorkflow.name_WF: (String)	expression

Failed at line 1, column 1: string matching regex `^\"*"\`$' expected but `e' found

Outputs

helloWorldWorkflow.helloWorldTask.Hello_World_OutputFile: (File)	expression

Failed at line 1, column 1: `workspace.' expected but `e' found

Referenced Method

Namespace:	birger	Created:	June 9, 2016 4:27 PM
Name:	helloworld	Entity Type:	Workflow
Snapshot ID:	4	Synopsis:	Hello World for Tool Developers Workshop

Documentation:

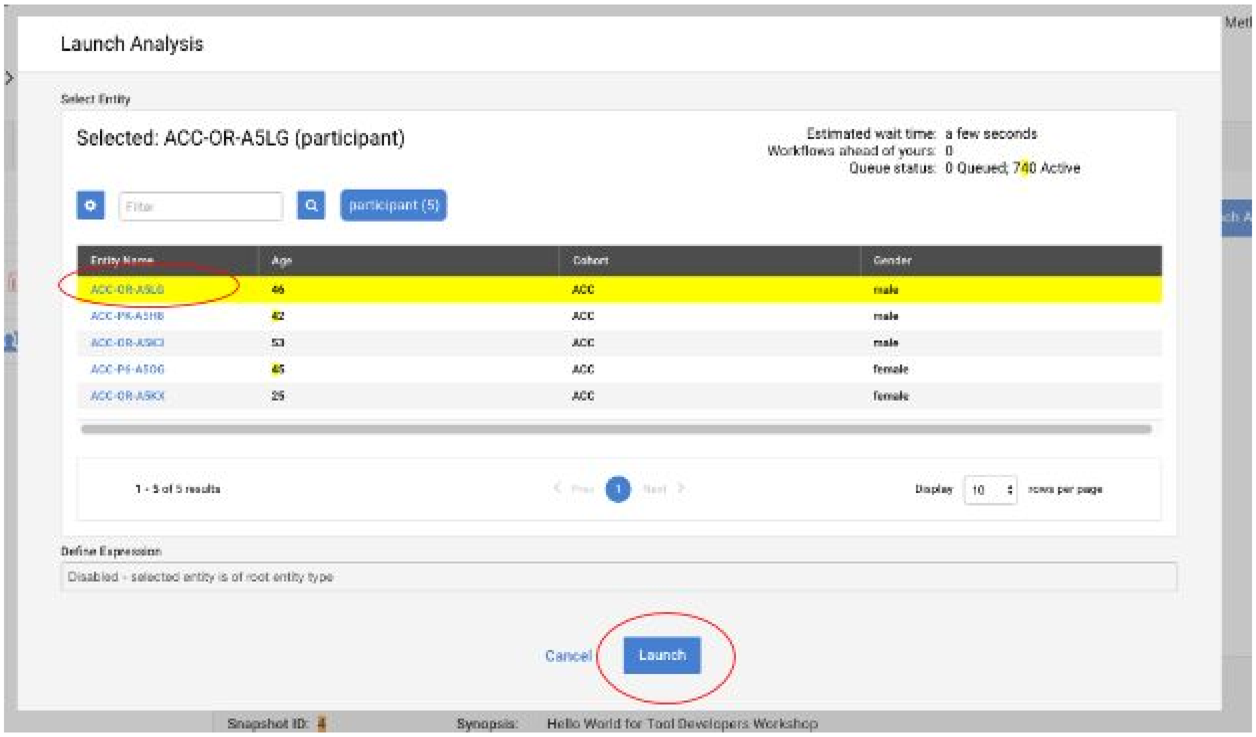
No documentation provided

WDL: Expand

- Specify expressions for the workflow's input and output parameters.
    - Set the expression for the `helloWorldWorkflow.name_WF` input paramter to `this.name`
    - Set the expression for the the `helloWorldWorkflow.helloWorldTask.Hello_World_OutputFile` to `this.greeting`
    - Click Save

4. Launch the helloworld method (workflow) on a single participant

- Click Launch Analysis…
- Select the first participant in the list and click Launch



Launch Analysis

Select Entity

Selected: ACC-OR-A5LG (participant)

Estimated wait time: a few seconds
Workflows ahead of yours: 0
Queue status: 0 Queued; 740 Active

participant (5)

Entity Name	Age	Cohort	Gender
ACC-OR-A5LG	46	ACC	male
ACC-PK-A5H8	42	ACC	male
ACC-OR-A5KO	53	ACC	male
ACC-P6-A5OG	45	ACC	female
ACC-OR-A5KX	25	ACC	female

1 - 5 of 5 results

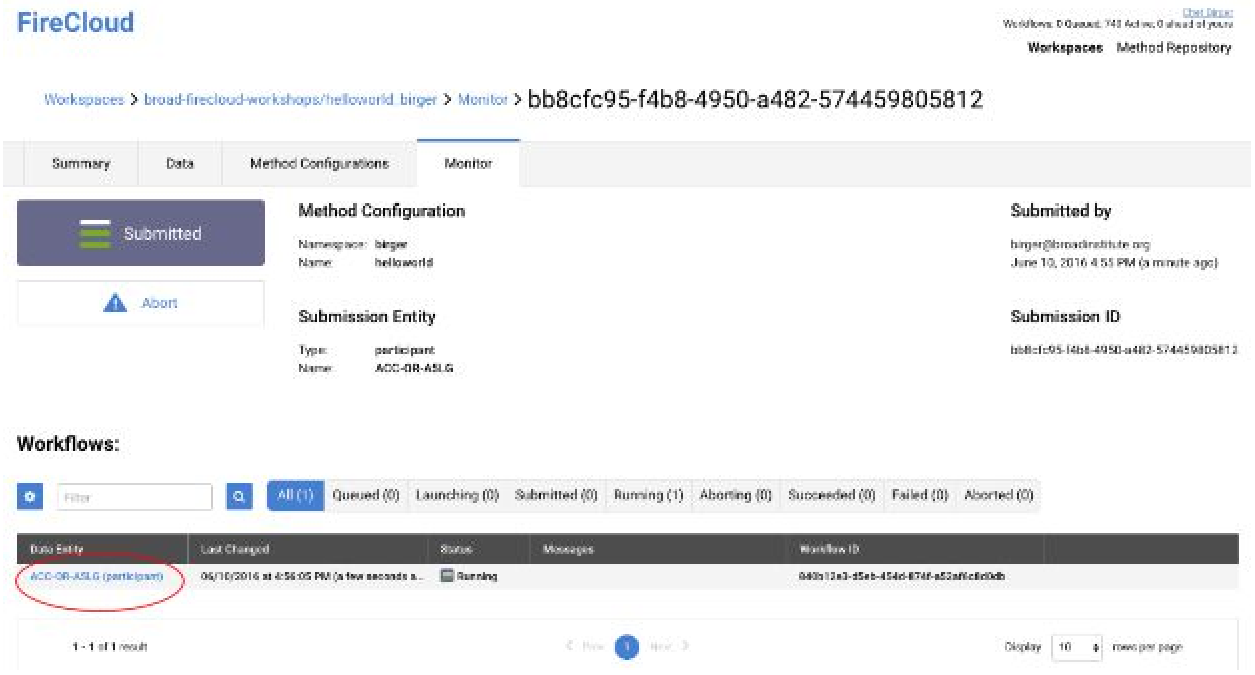
Display 10 rows per page

Define Expression

Disabled - selected entity is of root entity type

Snapshot ID: 4   Synopsis: Hello World for Tool Developers Workshop

- Monitor the running workflow
    - Click on the single workflow in the monitor tab's list of the workspace's submitted workflows to drill down into the workflow's status. Note that if the workflow's status is queued, you will not be able to drill down; if this is the case, refresh the page until the Status becomes "Running"



FireCloud

Workflows: 0 Queued; 740 Active; 0 ahead of yours

Workspaces > broad-firecloud-workshops/helloworld_birger > Monitor > bb8cfc95-f4b8-4950-a482-574459805812

Summary | Data | Method Configurations | Monitor

Submitted

Abort

Method Configuration

Namespace: birger
Name: helloworld

Submission Entity

Type: participant
Name: ACC-OR-A5LG

Submitted by

birger@broadinstitute.org
June 10, 2016 4:55 PM (a minute ago)

Submission ID

bb8cfc95-f4b8-4950-a482-574459805812

Workflows:

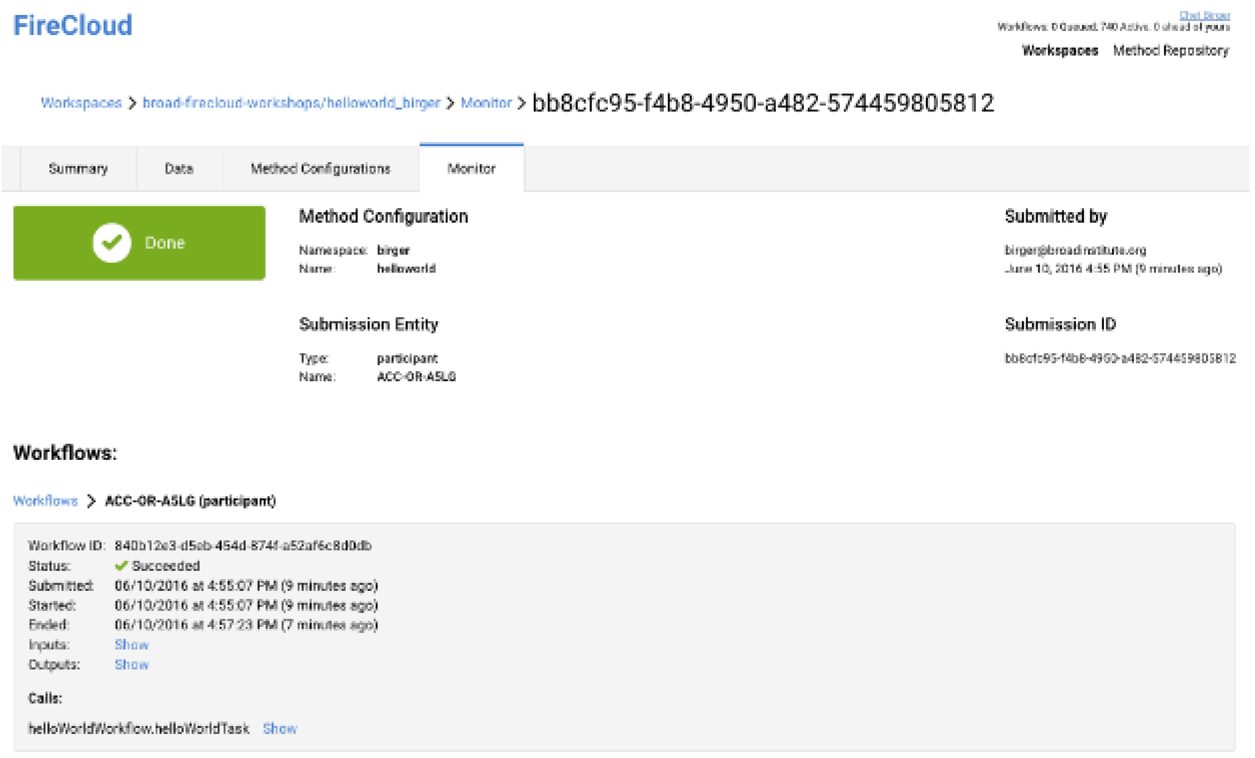
All (1) | Queued (0) | Launching (0) | Submitted (0) | Running (1) | Aborting (0) | Succeeded (0) | Failed (0) | Aborted (0)

Data Entity	Last Changed	Status	Messages	Workflow ID
ACC-OR-A5LG (participant)	06/10/2016 at 4:56:05 PM (a few seconds a...	Running		840b13e3-d5eb-454d-874f-a52af6c8d0db

1 - 1 of 1 result

Display 10 rows per page

- The HelloWorld workflow should complete within minutes.



Workspaces > broad-firecloud-workshops/helloworld_birger > Monitor > bb8cfc95-f4b8-4950-a482-574459805812

Method Configuration

Namespace: birger
Name: helloworld

Submitted by

birger@broadinstitute.org
June 10, 2016 4:55 PM (9 minutes ago)

Submission Entity

Type: participant
Name: ACC-OR-A5LG

Submission ID

bb8cfc95-f4b8-4950-a482-574459805812

Workflows:

Workflows > ACC-OR-A5LG (participant)

Workflow ID: 840b12e3-d5eb-454d-874f-a52af6c8d0db
Status: ✓ Succeeded
Submitted: 06/10/2016 at 4:55:07 PM (9 minutes ago)
Started: 06/10/2016 at 4:55:07 PM (9 minutes ago)
Ended: 06/10/2016 at 4:57:23 PM (7 minutes ago)
Inputs: Show
Outputs: Show

Calls:

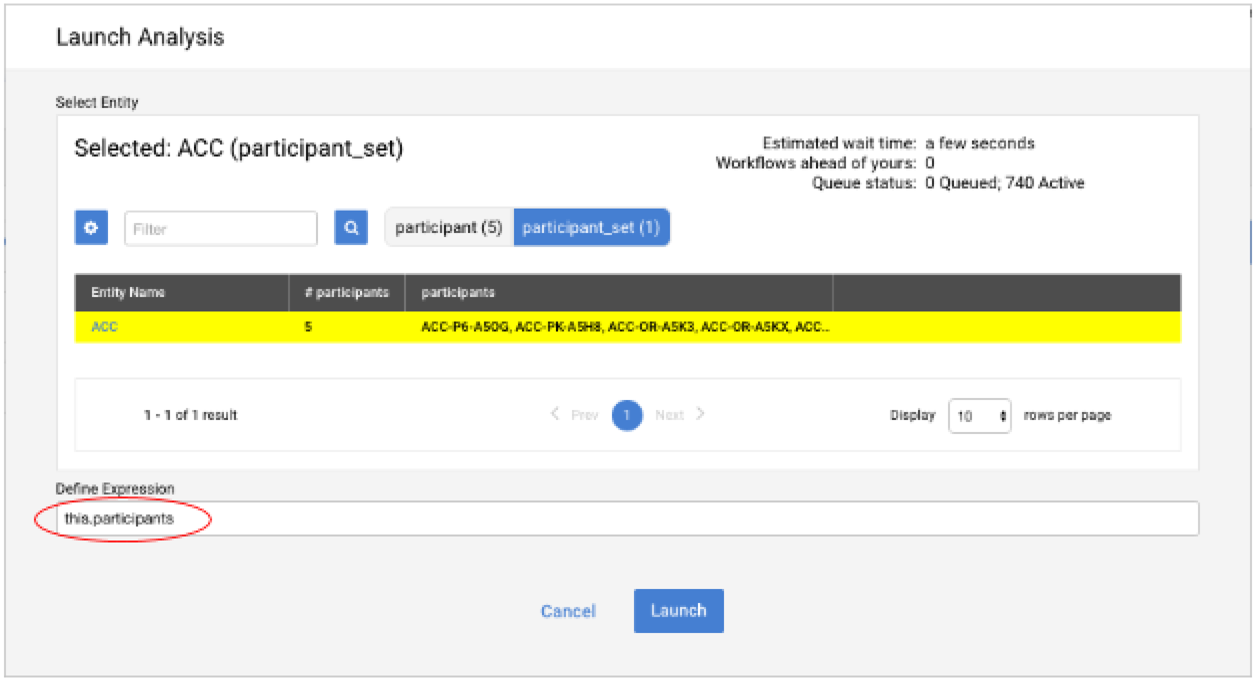
helloWorldWorkflow.helloWorldTask Show

5. View the results

- Go to the Data tab and click on the first participant's greeting attribute value
- Click Open

6. Launch the helloworld method (workflow) on all participants in a participant set

- Go to the Method Configuration Tab and select the helloworld method configuration
- Click on Launch Analysis…
- Click on participant_set(1)
- Select the participant set ACC (it will become highlighted in yellow)
- In the Define Expression box, enter `this.participants`



Launch Analysis

Select Entity

Selected: ACC (participant_set)

Estimated wait time: a few seconds
Workflows ahead of yours: 0
Queue status: 0 Queued; 740 Active

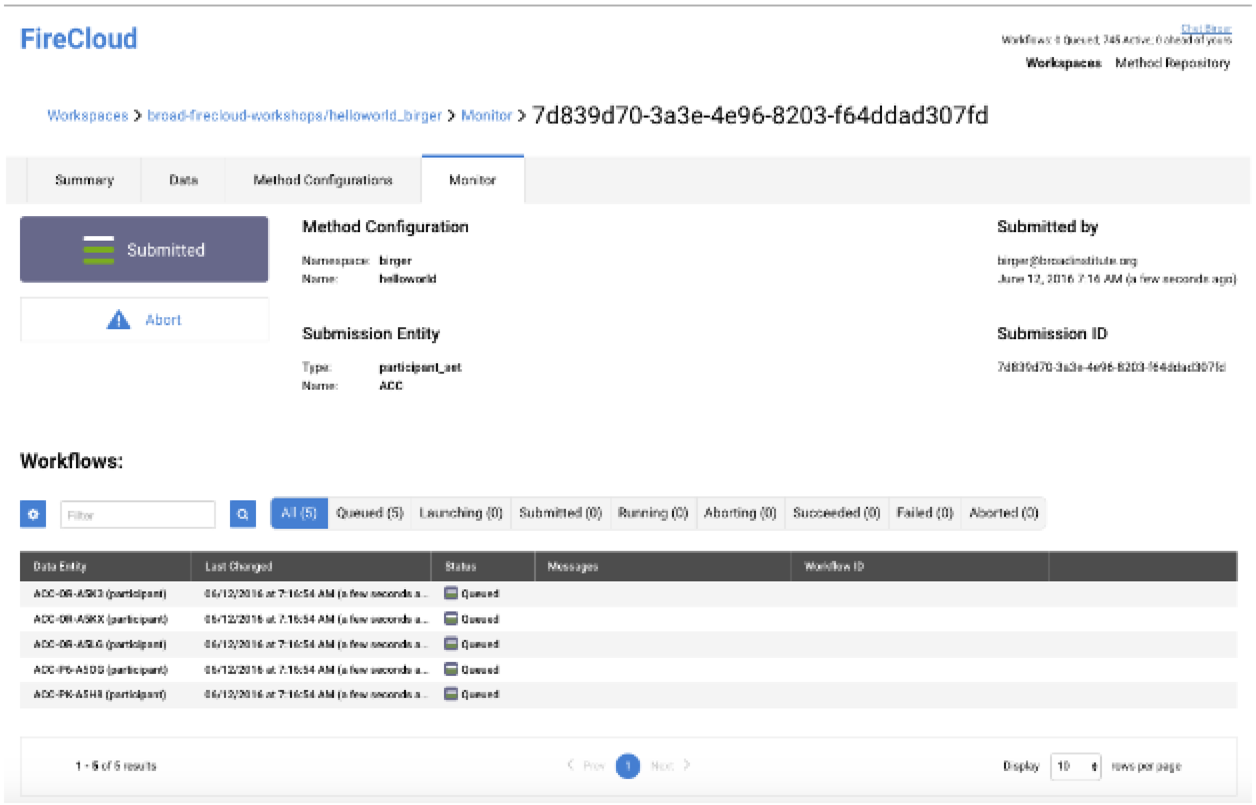
Entity Name	# participants	participants
ACC	5	ACC-P6-A5OG, ACC-PK-A5H8, ACC-OR-A5K3, ACC-OR-A5KX, ACC...

1 - 1 of 1 result

Define Expression

this.participants

- Click Launch
  In most cases, you will see a listing of the five workflows (one for each participant) that have been queued for submission.



Workspaces > broad-firecloud-workshops/helloworld_birger > Monitor > 7d839d70-3a3e-4e96-8203-f64ddad307fd

Method Configuration
Namespace: birger
Name: helloworld

Submitted by
birger@broadinstitute.org
June 12, 2016 7:16 AM (a few seconds ago)

Submission Entity
Type: participant_set
Name: ACC

Submission ID
7d839d70-3a3e-4e96-8203-f64ddad307fd

Workflows:

All (5) | Queued (5) | Launching (0) | Submitted (0) | Running (0) | Aborting (0) | Succeeded (0) | Failed (0) | Aborted (0)

Data Entity	Last Changed	Status	Messages	Workflow ID
ACC-OR-A5K3 (participant)	06/12/2016 at 7:16:54 AM (a few seconds a...	Queued		
ACC-OR-A5KX (participant)	06/12/2016 at 7:16:54 AM (a few seconds a...	Queued		
ACC-OR-A5LG (participant)	06/12/2016 at 7:16:54 AM (a few seconds a...	Queued		
ACC-P6-A5OG (participant)	06/12/2016 at 7:16:54 AM (a few seconds a...	Queued		
ACC-PK-A5H8 (participant)	06/12/2016 at 7:16:54 AM (a few seconds a...	Queued		

1 - 5 of 5 results

- Click on the Monitor tab, and then click on your most recent analysis submission. You will be able to see the current status of the five helloworld workflows.
- When all five workflows have the Done status, click on the data tab, and look at the particpants table. You will see a listing of the five output files. Click on any of the output file hyperlinks to see its content.

2. Linear Chaining WDL

The purpose of this exercise is to demonstrate how to create the WDL “plumbing” that feeds the output of task #1 to the output of task #2, whose output, in turn, is fed into task #3. This results in a three step workflow, where tasks #1 through #3 run sequentially. For expediency we will demonstrate this by running the workflow on the locally running version of cromwell rather than uploading and running it on FireCloud. The purpose of the exercise is to highlight how tasks outputs get fed into the inputs of downstream tasks, which may be done independently of FireCloud.

You will find the wdl file (linear_chain.wdl) in the linear_chain_wdl folder of the unzipped archive.

```
task addTwoTask {

       String inputNum

       command <<<
              OUT_NUM=$((${inputNum} + 2)) ;
              echo $OUT_NUM
              >>>

       output {
              String outNum=read_string(stdout())
              }

       runtime {
              docker: "ubuntu:14.04.4"
              }

       }

task multiplyByFiveTask {

       String inputNum

       command <<<
              OUT_NUM=$((${inputNum} * 5)) ;
              echo $OUT_NUM
              >>>
```

```
      output {
            String outNum=read_string(stdout())
            }

      runtime {
            docker: "ubuntu:14.04.4"
            }

      }


task addThreeTask {

      String inputNum

      command <<<
            OUT_NUM=$((${inputNum} + 3)) ;
            echo $OUT_NUM
            >>>

      output {
            String final=read_string(stdout())
            }

      runtime {
            docker: "ubuntu:14.04.4"
            }

      }

workflow calculatorWorkflow {

      String inputNum

      call addTwoTask {
            input:
                  inputNum=inputNum
            }

      call multiplyByFiveTask {
            input:
                  inputNum=addTwoTask.outNum
            }

      call addThreeTask {
            input:
                  inputNum=multiplyByFiveTask.outNum
            }


      }
```

There are several things to note with this wdl file:

- The workflow is trivial:
    - `y = x + 2`
    - `z = y * 5`
    - `q = z + 3`
- A command is a *task section* that starts with the keyword 'command', and is enclosed in curly braces or `<<< >>>.` Sometimes a command is sufficiently long enough or might use { characters that using a different set of delimiters would make it more clear. In this case, enclose the command in <<<...>>>
- Our docker image is an official base ubuntu image. Our command block is making a series of bash shell command calls; we are not running a java or python application that is bundled into a custom docker image. Don't focus on the bash commands...the important aspect of this file is how the output of one task is specified as the input of a downstream task.

2.1 Test linear_chain WDL running Cromwell locally

1. Create json template for workflow inputs

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/wdltool-0.4.jar inputs \
~/Workshop_Materials/linear_chain_wdl/linear_chain.wdl > linear_chain_inputs.json
WMDAF-281:test birger$ cat linear_chain_inputs.json
{
  "calculatorWorkflow.inputNum": "String"
}
WMDAF-281:test birger$
```

   Using your text editor, edit `linear_chain_inputs.json` and replace "String" above with a number; e.g.,

   `"calculatorWorkflow.inputNum": "1"`

2. Run WDL with local Cromwell

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/cromwell-0.19.jar run \
~/Workshop_Materials/linear_chain_wdl/linear_chain.wdl linear_chain_inputs.json
[2016-06-14 12:38:34,958] [info] Slf4jLogger started
[2016-06-14 12:38:35,11] [info] RUN sub-command
[2016-06-14 12:38:35,12] [info]   WDL file:
/Users/birger/Workshop_Materials/linear_chain_wdl/linear_chain.wdl
[2016-06-14 12:38:35,13] [info]   Inputs: linear_chain_inputs.json
...
...
[2016-06-14 12:38:44,927] [info] WorkflowActor [984e68d9]: Beginning transition from Running
to Succeeded.
[2016-06-14 12:38:44,939] [info] WorkflowActor [984e68d9]: transitioning from Running to
```

```
Succeeded.
{
  "calculatorWorkflow.addThreeTask.final": "18",
  "calculatorWorkflow.addTwoTask.outNum": "3",
  "calculatorWorkflow.multiplyByFiveTask.outNum": "15"
}
[2016-06-14 12:38:44,958] [info] SingleWorkflowRunnerActor workflow finished with status
'Succeeded'.
WMDAF-281:test birger$
```

Note that the outputs from each task in the workflow are displayed in a json object written to stdout.

3. Task Aliasing

You often will want to call the same task multiple times within a workflow. You will need to be able to distinguish between the different calls when referencing outputs. This is done through *task aliasing*. The following exercise provides a simple example of aliasing.

```
               input:
        task addTwoTask {

        String inputNum

        command <<<
               OUT_NUM=$((${inputNum} + 2)) ;
               echo $OUT_NUM
               >>>

        output {
               String outNum=read_string(stdout())
               }

        runtime {
               docker: "ubuntu:14.04.4"
               }
        }

task addNumsTask {

        String inputNumOne
        String inputNumTwo

        command <<<
               OUT_NUM=$((${inputNumOne} + ${inputNumTwo})) ;
               echo $OUT_NUM > outFile.txt
               echo "The numbers $IN_NUM_ONE and $IN_NUM_TWO were added!" > log.txt
               >>>

        output {
               File outFile="outFile.txt"
               String outNum=read_string("outFile.txt")
               File log="log.txt"
               }

        runtime {
               docker: "ubuntu:14.04.4"
```

```
                }
        }

workflow TaskAliasingWorkflow {

        String dataOne
        String dataTwo

        call addTwoTask as FirstAdder {
                input:
                        inputNum=dataOne
                }


        call addTwoTask as SecondAdder {
                input:
                        inputNum=dataTwo
                }

        call addNumsTask {
                input:
                        inputNumOne=FirstAdder.outNum,
                        inputNumTwo=SecondAdder.outNum
                }

}
```

Note the following:

- The workflow is a sequence of trivial arithmetic operations:
    - y1 = x1 + 2
    - y2 = x2 + 2
    - z = y1 + y2
- As with the linear chaining example, the command statements are enclosed in <<< >>> rather than curly braces, the command block is a series of bash shell commands and the loaded docker image is an official base ubuntu image.
- The two calls to addTwoTask are independent of one another (they are not wired together); cromwell can schedule the two task instances to run in parallel.
- The addNumsTask has two input parameters and produces three outputs: a String representation of the sum and two files, a log file and the file containing the sum.

3.1 Test task_aliasing WDL running Cromwell locally

1. Create json template for workflow inputs

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/wdltool-0.4.jar inputs \
~/Workshop_Materials/task_aliasing_wdl/task_aliasing.wdl > task_aliasing_input.json
WMDAF-281:test birger$ cat task_aliasing_input.json
{
  "TaskAliasingWorkflow.dataOne": "String",
  "TaskAliasingWorkflow.dataTwo": "String"
}
WMDAF-281:test birger$
```

Using your text editor, edit task_aliasing_input.json and replace `Strings` above with numbers; e.g.,

```
"TaskAliasingWorkflow.dataOne": "1"
"TaskAliasingWorkflow.dataTwo": "5"
```

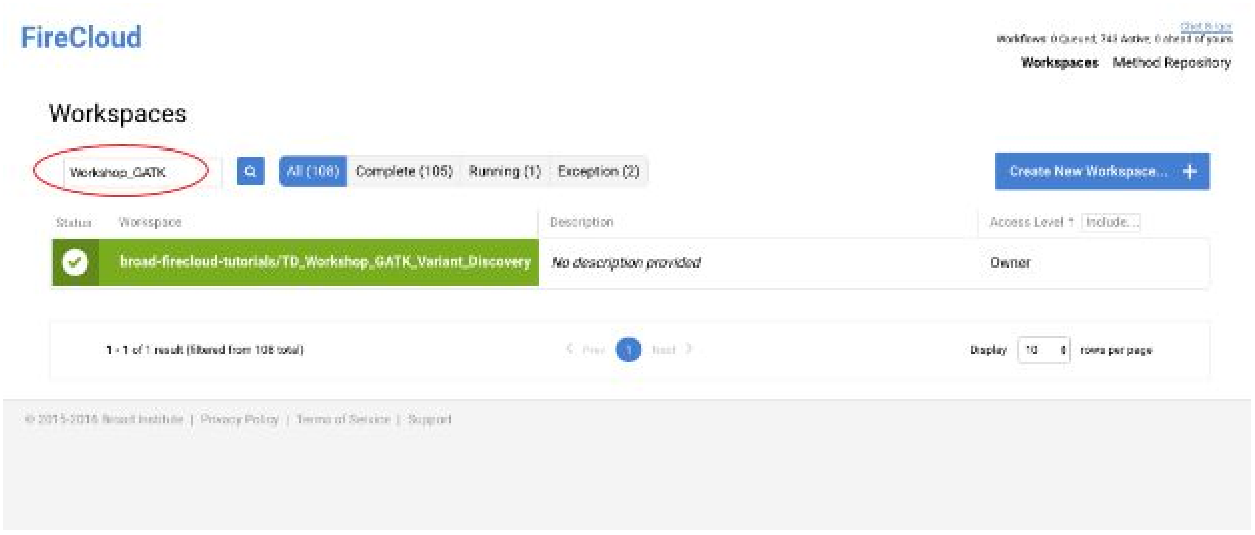
2. Run WDL with local Cromwell

```
WMDAF-281:test birger$ java -jar ~/Workshop_Materials/bin/cromwell-0.19.jar run \
~/Workshop_Materials/task_aliasing_wdl/task_aliasing.wdl task_aliasing_input.json
[2016-06-14 12:45:10,660] [info] Slf4jLogger started
[2016-06-14 12:45:10,718] [info] RUN sub-command
[2016-06-14 12:45:10,719] [info]   WDL file:
/Users/birger/Workshop_Materials/task_aliasing_wdl/task_aliasing.wdl
[2016-06-14 12:45:10,720] [info]   Inputs: task_aliasing_input.json
...
...
[2016-06-14 12:48:23,751] [info] WorkflowActor [f22a518e]: Beginning transition from Running
to Succeeded.
[2016-06-14 12:48:23,760] [info] WorkflowActor [f22a518e]: transitioning from Running to
Succeeded.
{
  "TaskAliasingWorkflow.SecondAdder.outNum": "7",
  "TaskAliasingWorkflow.addNumsTask.outFile":
"/Users/birger/test/cromwell-executions/TaskAliasingWorkflow/f22a518e-244e-4700-a8e0-fdc8c80
babb5/call-addNumsTask/outFile.txt",
  "TaskAliasingWorkflow.addNumsTask.outNum": "10",
  "TaskAliasingWorkflow.addNumsTask.log":
"/Users/birger/test/cromwell-executions/TaskAliasingWorkflow/f22a518e-244e-4700-a8e0-fdc8c80
babb5/call-addNumsTask/log.txt",
  "TaskAliasingWorkflow.FirstAdder.outNum": "3"
}
[2016-06-14 12:48:23,783] [info] SingleWorkflowRunnerActor workflow finished with status
'Succeeded'.
WMDAF-281:test birger$
```

Note the three outputs of the addNumsTask: outFile, outNum and log. The output files are written to the host file system. If running this workflow in FireCloud, the output files are written to the workspace's Google Cloud Storage bucket.

4. Follow Along Demonstration of Simple Variant Discovery Mini-Pipeline

4.1 Open a FireCloud session and filter workspace listings on Workshop_GATK



Status	Workspace	Description	Access Level
✓	broad-firecloud-tutorials/TD_Workshop_GATK_Variant_Discovery	*No description provided*	Owner

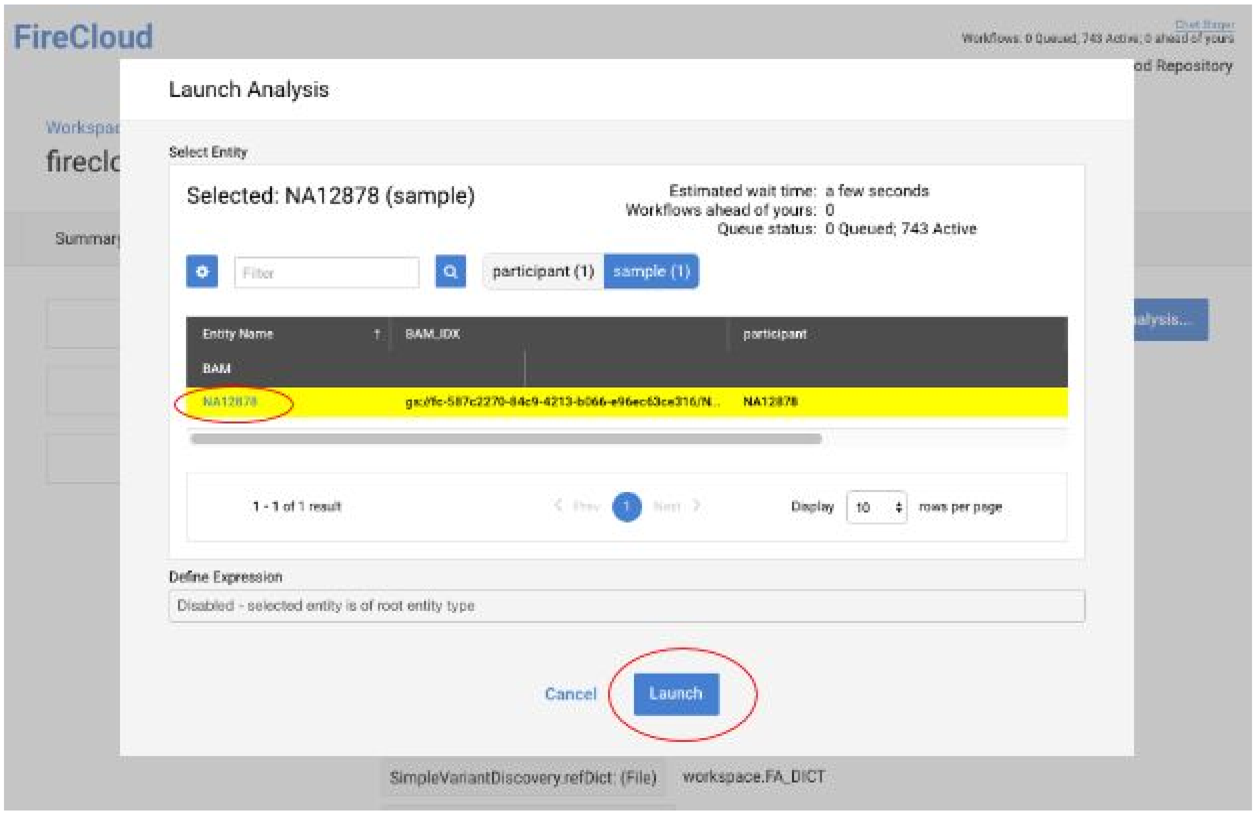
1 - 1 of 1 result (filtered from 108 total)

4.2 Enter the workspace broad-firecloud-tutorials/TD_Workshop_GATK_Variant_Discovery and clone

- Be sure to give the clone a unique name; e.g., broad-firecloud-workshops/TD_Workshop_GATK_Variant_Discovery_<your name>

4.3 Launch analysis

1. Go to the Method Configuration tab and select `simpleVariantDiscovery`
2. Click Launch Analysis…
3. Select the sample NA12878, and click Launch



Selected: NA12878 (sample)

Estimated wait time: a few seconds
Workflows ahead of yours: 0
Queue status: 0 Queued; 743 Active

Entity Name	BAM_IDX	participant
NA12878	gs://fc-587c2270-84c9-4213-b066-e96ec63ce316/N...	NA12878

1 - 1 of 1 result

Display 10 rows per page

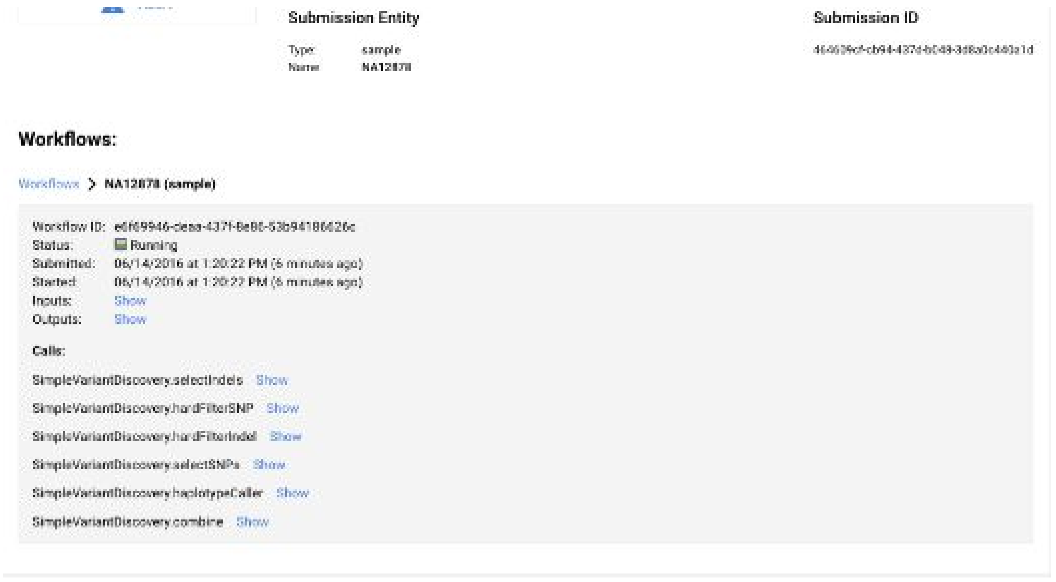
Define Expression

Disabled - selected entity is of root entity type

SimpleVariantDiscovery.refDict: (File) workspace.FA_DICT

4.4 Monitor status of jobs in launched workflow

1. Go to Monitor tab and select analysis submission
2. Select single sample workflow



Submission Entity

Type: sample
Name: NA12878

Submission ID

464639cf-cb94-437d-b049-3d8a0c440a1d

Workflows:

Workflows > NA12878 (sample)

```
Workflow ID:  e6f69946-cesa-437f-8e86-53b94186626c
Status:       Running
Submitted:    06/14/2016 at 1:20:22 PM (6 minutes ago)
Started:      06/14/2016 at 1:20:22 PM (6 minutes ago)
Inputs:       Show
Outputs:      Show

Calls:
SimpleVariantDiscovery.selectIndels   Show
SimpleVariantDiscovery.hardFilterSNP   Show
SimpleVariantDiscovery.hardFilterIndel   Show
SimpleVariantDiscovery.selectSNPs   Show
SimpleVariantDiscovery.haplotypeCaller   Show
SimpleVariantDiscovery.combine   Show
```

SUPPLEMENTARY Material:

1. How to set up a FireCloud Billing Project

In order for a FireCloud Administrator to create a new FireCloud Billing Project, you must first:

1. Set up a Google Billing Account, using one of the options below:

- Create your own Google Billing Account using a credit card or bank account.
- Talk to your institutional procurement office and see if they have a preferred account set up method with Google (such as a third party reseller or an existing account).
- Set up a Google Billing Account through a third party reseller. There are many options and two examples are: **Onix Networking** or **Sada Systems**. Third party resellers provide additional billing options at no extra cost.

2. Add **Google@broadinstitute.com** as a Billing Administrator to your Google Billing Account.

3. Request a new FireCloud Billing Project by submitting a **FireCloud Billing Project Request Form**. This request should include your Google Billing Account ID. You can can remove **Google@broadinstitute.com** as a Billing Administrator after you receive confirmation that your FireCloud Billing Project has been created.

Go **here** for more information. You can also email Help@FireCloud.org if you have any questions or problems.

2. Run ‘Workshop_MiniMutationCalling’ via the command line

Note: this does not include the steps to upload TSVs.

```
wm8b1-75c:May24Workshop esalinas$ cat mini_mutation_calling_script.sh
#!/bin/bash

#set verbosity
set -x

#define paths for software
FISSFC_PATH=/usr/local/bin/fissfc
PYTHON_PATH=/usr/local/bin/python

####################################
# REQUIREMENTS
# 1) Google Cloud SDK is required https://cloud.google.com/sdk/
# 2) Having logged in with 'gcloud auth login'
# (using the google cloud SDK) is also required
# 3) fissfc is required (https://pypi.python.org/pypi/fissfc/0.6.0)
# 4) Python required to have a compatible and sufficiently recent openssl
#                 OpenSSL 1.0.2e 3 Dec 2015 or newer should suffice
echo -ne "import ssl\nprint(ssl.OPENSSL_VERSION)"|$PYTHON_PATH
#This script was developed using OpenSSL 1.0.2e 3 Dec 2015
#With some older versions of OpenSSL an error
#                 'httplib2.SSLHandshakeError:
#                 [SSL: TLSV1_ALERT_PROTOCOL_VERSION]
#                 tlsv1 alert protocol version (_ssl.c:590)' may arise

#test connectivity
$FISSFC_PATH ping
#2016-05-18T15:11:08.220+0000
CONN=`/usr/local/bin/fissfc ping 2>/dev/null |perl -ne 'print "success"  if
/^\d{4}\-\d{2}\-[A-Za-z0-9]{5}:\d+:\d+\.\d+\+\d+$/'|tail -1` ;
if [ "$CONN" = "success" ] ; then
        echo "Ping seems to have been successful!" ;

        #list workspaces
        $FISSFC_PATH space_list

        #define workspace names and namespaces
        TUTORIAL_WSNS="broad-firecloud-tutorials"
        TUTORIAL_WS="Workshop_MiniMutationCalling"
        TARGET_WSNS="broad-firecloud-testing"
        TARGET_WS="eddie_mmc2"

        #clone the mini mutation workspace
        echo -ne "To clone $TUTORIAL_WSNS / $TUTORIAL_WS to $TARGET_WSNS /
$TARGET_WS ..." ;
        $FISSFC_PATH space_clone $TUTORIAL_WSNS $TUTORIAL_WS $TARGET_WSNS
$TARGET_WS ;

        #list data in the workspace
        echo -ne "import firecloud.api as
fapi\nr,c=fapi.get_entities_with_type(\"$TARGET_WSNS\",\"$TARGET_WS\")\nprint c"|
$PYTHON_PATH

        #submit a job!
        #against the HCC_pairs
        echo -ne "import firecloud.api as
fapi\nfapi.create_submission(\"$TARGET_WSNS\",\"$TARGET_WS\",\"workshop\",\"MiniMutati
onCalling_Cfg\",\"HCC1954_100_gene_pair\",\"pair\",None)"|$PYTHON_PATH ;
        echo -ne "import firecloud.api as
fapi\nfapi.create_submission(\"$TARGET_WSNS\",\"$TARGET_WS\",\"workshop\",\"MiniMutati
onCalling_Cfg\",\"HCC1143_WE_pair\",\"pair\",None)"|$PYTHON_PATH ;
        #polling, polling, polling....for submission status!
        DONE_COUNT=0
        until [ "$DONE_COUNT" == "2" ] ; do
                TOKEN=`gcloud auth print-access-token`
                echo "Sleep a bit, then poll for submission status...." ;
                sleep 30
                DONE_COUNT=`curl -X GET --header "Accept: application/json" --header
"Authorization: Bearer $TOKEN"
"https://api.firecloud.org/api/workspaces/${TARGET_WSNS}/${TARGET_WS}/submissions"
2>/dev/null |grep -i 'status'|grep -ic 'done'`
                echo "DONE COUNT IS $DONE_COUNT" ;
                date
        done ;

        #list data in the workspace
        echo -ne "import firecloud.api as
fapi\nr,c=fapi.get_entities_with_type(\"$TARGET_WSNS\",\"$TARGET_WS\")\nprint c"|
$PYTHON_PATH

else
```

```
        echo "Ping seems to have been unsuccessful... abort!" ;
fi ;
```

3. Glossary

Basic Concepts

Analysis Submission
A user submits an Analysis Submission to the workspace service when launching a method configuration against an entity set. An analysis submission is a combination of a method config and entity set; this combination identifies the method that will run, the number of times it will be run (the number of entities in the set), and the inputs and outputs for each run.

BAM file
An input unit consisting of tab-delimited text that contains sequence alignment data. It is the binary version of a SAM file.

Controlled Access
De-identified data that may be unique to individuals. FireCloud users with dbGaP-authorization and a linked eRA Commons account can access TCGA controlled access data.

Data Model
Organizes data and meta-data for workspaces and analysis runs. The data model includes predefined entity types (e.g., participants and sample sets), relationships, and entity attributes. For your convenience, results from analysis runs are populated directly to the data model. Currently, the data model is tailored to TCGA data, but will be extensible to non-TCGA projects with a germline or cell-line focus.

Entity
Refers to physical items (e.g., participants) or collections of physical items (e.g., participant sets). Entities provide organization and hierarchical structure for data. For example, a participant entity refers to a participant. A sample entity refers to a sample that may belong to that participant.

Entity Attributes
FireCloud uses entity attributes to describe data entities (e.g., a participant identifier) and reference entity file locations (e.g., the URL to a Google Cloud Storage bucket). Entity attributes can be fed into and populated from a workflow analysis.

FireCloud RESTful API
All functionality presented through the user interface is also available to users through a public-facing secure RESTful API. Comprehensive on-line documentation for this API is

available at https://api.firecloud.org. This online documentation employs the Swagger representation of RESTful APIs. The FireCloud RESTful API's endpoints are organized into the following categories:

- Entities
- Method Configurations
- Method Repository
- NIH
- OAuth
- Profile
- Storage
- Submissions
- Workspaces

Load Files (TSV Files)
FireCloud uses tab-separated-value (TSV) files to import entities and entity attributes into the Data tab. Each line in the TSV file corresponds to an entity and must reference entities of the same type. The FireCloud Data Model supports the following entity types:

- Participant
- Sample
- Pair
- Participant Set
- Sample Set
- Pair Set

Methods
A WDL description of a task or workflow in FireCloud.

Method Configurations (Method Configs)
Bind data to Methods and specify which attributes to use as inputs and outputs to an analysis runs. You can specify attributes in Method Config output fields that will get updated with results from an analysis run.

Method Repository
Contains methods for analyzing data (workflows and their constituent tasks), and method configs. Tool developers can upload their own methods using the FireCloud Command Line Interface (CLI).

Open Access
Public de-identified data that *is not* unique to individuals. All FireCloud users can access open access TCGA data.

Task

In FireCloud methods and WDL, tasks refer to executable programs that are bundled into a Docker image.

Workflow
Workflows are comprised of one or more tasks and contain the method and the method input parameters. FireCloud submits both tasks and workflows to Google Job Execution System (JES) when you run an analysis.

Workspace
Computational sandbox in which a FireCloud user organizes genomic data and metadata into a data model.

Workspace Access Controls (ACLs)
Define permissions and enable the secure sharing of workspaces among FireCloud users. ACLs contain three access levels: READER, WRITER, and OWNER where each access level represents an expanded set of permissions from the previous.

Workspace Attributes
Globally accessible input values within a workspace. If you enter workspace attributes in the workspac**e Summary** tab, they can serve as inputs for any Method Config within your workspace.

Google Cloud Platform Concepts

Google Billing Account
In order for a FireCloud Administrator to create a new FireCloud Google Project, you must first create a Google Billing Account. Google Billing Accounts are billed for cloud storage and compute costs that are tracked through FireCloud Google Projects. You will need to provide a bank account or credit card to set up a Google Billing Account, or use a Google Reseller for alternative payment options (e.g., cost objects).

Google Cloud Storage Bucket
Each workspace is associated with a single dedicated Google Cloud Storage bucket, created under the Google Project with which the workspace is created.

Google Developers Console
The Google Developers Console is the user interface for Google Cloud Platform. You can view buckets and bucket data and Google Project information through the Google Developers Console.

Google Project
Every workspace is linked to a single Google Project that tracks all cloud storage and cloud

compute costs incurred within that workspace. Only FireCloud administrators can create and grant you access to Google Projects for use in FireCloud.

GSUTIL
Google Cloud Storage's command line utility. Use this to upload data and files to Google buckets.

Tool Developer Concepts

Cromwell
Cromwell is the workflow execution service used to run and test WDL workflows. When creating WDL workflows, you can test on a local installation of the Cromwell execution engine prior to uploading and testing on FireCloud. Cromwell reads WDL, which describes executable tasks packaged into docker containers. Cromwell then calls Google's Job Execution System to run the executable tasks packaged into docker containers.

Docker
FireCloud uses Docker to distribute tools and applications for use in its methods. Docker allows applications and their dependencies to be packaged into discrete runtime environments, called Docker containers.

Docker Container
Docker containers wrap software in a file system that can contain the dependencies to run your tools on FireCloud. These dependencies can include code, system tools, system libraries and anything you can install on a server, thus enabling portability of tools across operating systems.

Docker Host
Virtual machine on which containers are launched, managed with 'docker-machine.'

DockerHub
DockerHub is a cloud-based registry service for Docker images. You can store and share your Docker images through repositories (repos), both public and private for use on FireCloud.

Docker Image
Docker images store software and operating systems.

FireCloud Command Line Interface (CLI)
This FireCloud CLI enables tool developers to push methods and method configurations to FireCloud.

FireCloud Cookbook

A newly developed tool that allows users to run most processes available in the FireCloud user interface through the command line. You can run Hands-on exercises from the workshop using this tool.

FISSfc
Contains bindings to the FireCloud RESTful API and allows users to script FireCloud tasks through the command line, bypassing the FireCloud user interface.

WDL (Workflow Description Language)
Workflow Description Language (WDL) is a language specifically designed for expressing genomics workflows. WDL workflows are represented in a way that can be read by humans and understood by Cromwell, the Workflow Execution Service that will run the specified tools to analyze data.