LA-UR-20-22721
Approved for public release; distribution is unlimited.

Title:	BEE - FY20 P6-3: Release BEEWorkflowManager, BEETaskManager, and client application 2.3.6.01 – LANL ATDM ST / STNS01-4 P6 Milestone Completion Documentation
Author(s):	Randles, Timothy C.
Intended for:	Exascale Computing Project reporting requirement
Issued:	2020-04-03

BEE - FY20 P6-3: Release BEEWorkflowManager, BEETaskManager, and client application

2.3.6.01 – LANL ATDM ST / STNS01-4 P6 Milestone Completion Documentation

Tim Randles
for the BEE Team

April 1, 2020

NNSA
National Nuclear Security Administration
Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

Activity Description

Release BEEWorkflowManager, BEETaskManager, and client software

The BEEWorkflowManager daemon runs on the HPC cluster login node. It accepts workflows submitted by the BEE client. These workflows are specified using the Common Workflow Language (CWL) standard. The BEEWorkflowManager loads workflows into the Neo4j graph database to create the workflow directed acyclic graph (DAG), and submits the workflow tasks to the BEETaskManager for execution. The BEEWorkflowManager records the state of the workflow and its tasks, and communicates this state to the BEE client. The BEEWorkflowManager will start, pause, and cancel a running workflow and its tasks at the command of the BEE client.

The BEETaskManager daemon runs on the HPC cluster login node. It accepts tasks from the BEEWorkflowManager, turns those tasks into HPC resource manager jobs (e.g. a slurm job script), and submits the job to the cluster resource manager. The BEETaskManager then tracks the status of the job (pending, running, complete) and updates the BEEWorkflowManager. The BEETaskManager will also cancel a queued or running job when commanded to do so by the BEEWorkflowManager. The first release of the BEETaskManager will support the Slurm resource manager and the Charliecloud linux container runtime.

Execution Plan

To complete this activity we will:

- implement a simple client command line interface for users. This command line interface will be the primary manner in which a user will submit workflows, control workflow execution, and monitor workflow status.
- implement the BEEWorkflowManager. The BEEWorkflowManager will be a daemon that accepts workflows from the user (via the command line client), parses the workflow, builds a workflow graph using the neo4j graph database, and submits ready workflow tasks to the BEETaskManager
- develop a BEETaskManager daemon that will accept workflow tasks from the BEEWorkflowManager
- develop a system whereby the BEETaskManager is able to format tasks into job scripts appropriate for submission to an HPC resource manager (Slurm is the initial target)
- develop functionality to deploy containerized HPC applications as part of the workflow task
- submit, control, and monitor workflow tasks as HPC jobs on behalf of the user

Completion Criteria

This activity will be complete when the BEE workflow engine can successfully perform the following functionality on a production Slurm HPC Cluster at LANL.

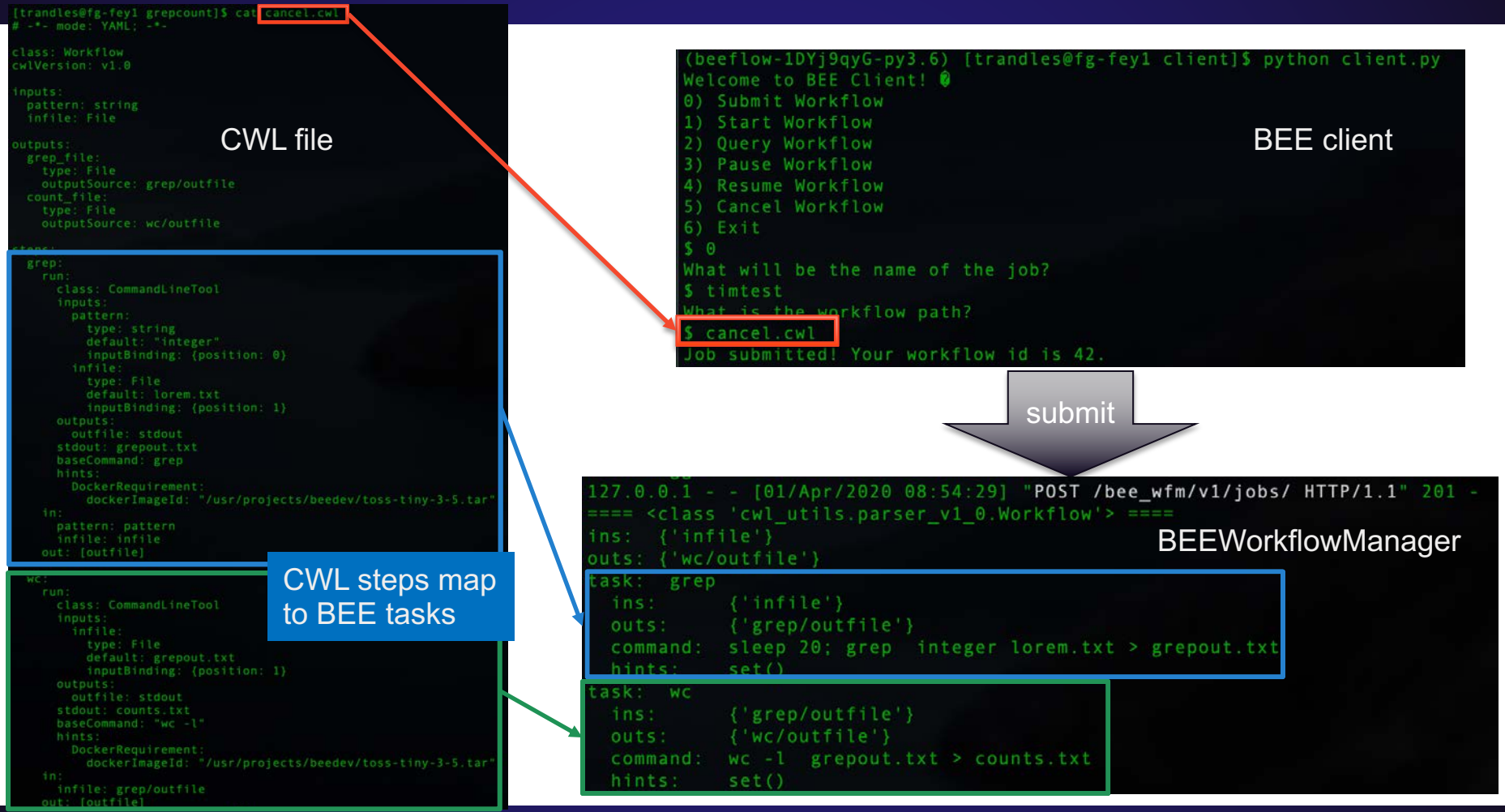
1. Accept a CWL workflow from the BEE client
2. Load the CWL workflow into a Neo4j graph database
3. Start/pause/cancel an active workflow
4. Submit ready tasks to the BEETaskManager
5. Report back to the BEE client the status of the submitted workflow and its tasks
6. BEETaskManager can accept a task from the BEEWorkflowManager
7. Format the accepted task as a Slurm job script
8. Use the Charliecloud linux container runtime to execute the task in the Slurm job
9. Submit the Slurm job to the HPC cluster
10. Report back to the BEEWorkflowManager the status of the submitted job
11. Cancel a submitted but not yet completed job when commanded to do so by the BEEWorkflowManager

Production Platform

- LANL HPC Production cluster named Fog
  - small 32-node production cluster used for taking new technologies “the last mile” from R&D to production
  - NNSA CTS-1 hardware
    - 2x Intel Broadwell CPUs (36 cores)
    - 256GB RAM
    - Intel OmniPath interconnect
  - LANL production software components (as of April 1, 2020)
    - TOSS 3 operating system (v3.5-2)
    - Slurm workload manager (v19.05.5)
    - Charliecloud HPC container runtime (v0.13)

Completion Criteria

1. Accept a CWL workflow from the BEE client

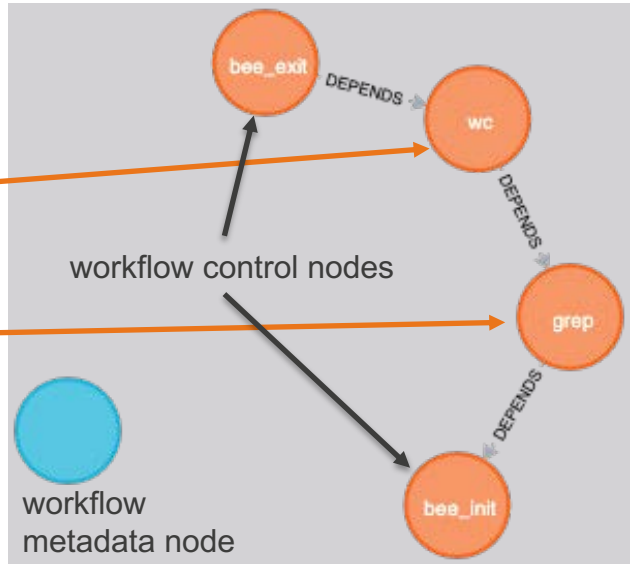


2. Load the CWL workflow into a Neo4j graph database

The BEEWorkflowManager enters the tasks into the Neo4j graph database. The database uses the task.ins and task.outs to determine the task dependencies. The ”bee_init” and “bee_exit” control nodes are used by the BEEWorkflowManager to control the beginning and ending of a workflow’s execution. They are automatically created by the BEEWorkflowManager and added to all workflows.

```
127.0.0.1 - - [01/Apr/2020 08:54:29] "POST /bee_wfm/v1/jobs/ HTTP/1.1" 201 -
==== <class 'cwl_utils.parser_v1_0.Workflow'> ====
ins:  {'infile'}
outs: {'wc/outfile'}
task:  grep
  ins:      {'infile'}
  outs:     {'grep/outfile'}
  command:  sleep 20; grep  integer lorem.txt > grepout.txt
  hints:    set()
task:  wc
  ins:      {'grep/outfile'}
  outs:     {'wc/outfile'}
  command:  wc -l  grepout.txt > counts.txt
  hints:    set()
```

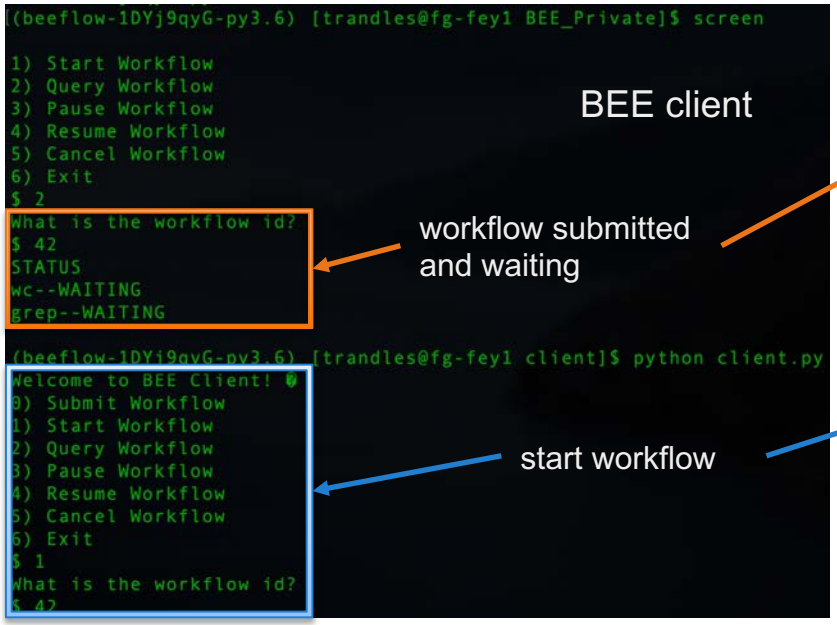
BEEWorkflowManager



Neo4j Graph

The workflow metadata node contains information such as the workflow requirements and hints contained in the CWL file.

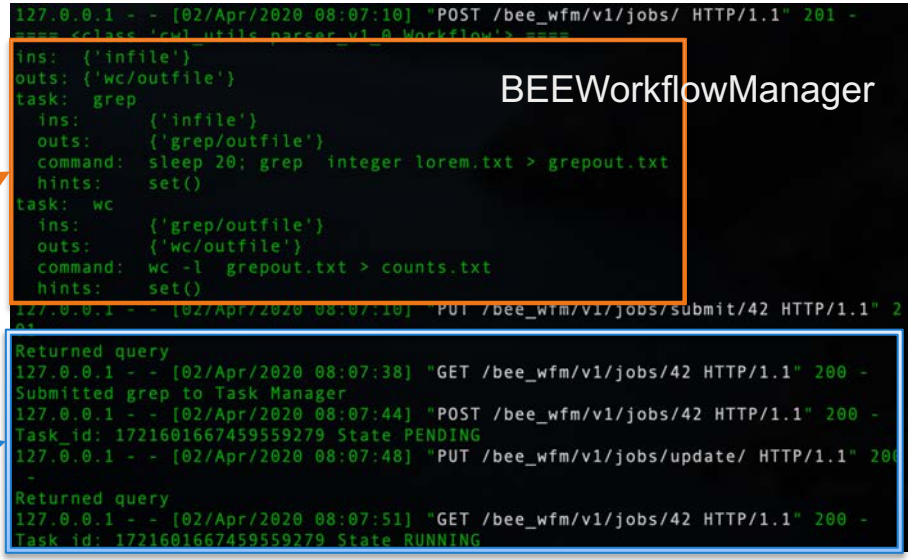
3. START/pause/cancel an active workflow



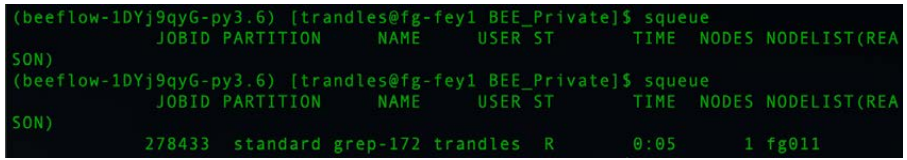
BEE client

workflow submitted and waiting

start workflow



BEEWorkflowManager



Slurm job starts running

3. start/PAUSE/cancel an active workflow

```
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 client]$ python client.py
Welcome to BEE Client!
0) Submit Workflow
1) Start Workflow
2) Query Workflow
3) Pause Workflow
4) Resume Workflow
5) Cancel Workflow
6) Exit
$ 1
What is the workflow id?
$ 42
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 client]$ python client.py
Welcome to BEE Client!
0) Submit Workflow
1) Start Workflow
2) Query Workflow
3) Pause Workflow
4) Resume Workflow
5) Cancel Workflow
6) Exit
$ 3
What is the workflow id?
$ 42
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 client]$ python client.py
Welcome to BEE Client!
0) Submit Workflow
1) Start Workflow
2) Query Workflow
3) Pause Workflow
4) Resume Workflow
5) Cancel Workflow
6) Exit
$ 2
What is the workflow id?
$ 42
STATUS
wc--WAITING
grep--COMPLETED
```

BEE client

start submitted workflow

pause running workflow

2nd task waiting

1st task completed

When the user pauses a running workflow in BEE, any running tasks will be allowed to run to completion but NO NEW tasks will be started until the workflow is resumed. You can see this by following the BEEWorkflowManager output below.

```
Submitted grep to Task Manager
127.0.0.1 - - [02/Apr/2020 08:26:43] "POST /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 1721601667459559279 State PENDING
127.0.0.1 - - [02/Apr/2020 08:26:44] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200
-
Task_id: 1721601667459559279 State RUNNING
127.0.0.1 - - [02/Apr/2020 08:26:48] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200
-
Workflow Paused
127.0.0.1 - - [02/Apr/2020 08:26:51] "PATCH /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 1721601667459559279 State COMPLETED
Saving wc
127.0.0.1 - - [02/Apr/2020 08:27:08] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200
-
Returned query
Submitted wc to Task Manager
Workflow Resumed
127.0.0.1 - - [02/Apr/2020 08:33:15] "PATCH /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 9088320184746133781 State PENDING
127.0.0.1 - - [02/Apr/2020 08:33:18] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200
-
Task_id: 9088320184746133781 State COMPLETED
Workflow Completed!
```

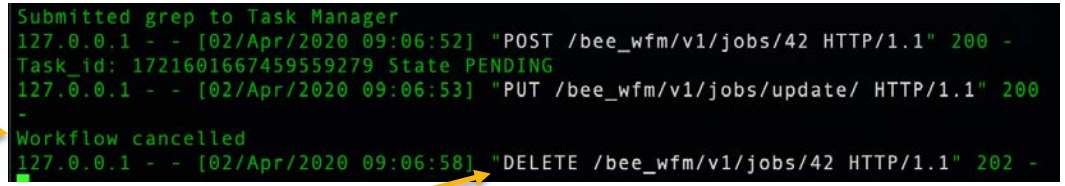
workflow resumes and finishes
(client command not shown)

3. start/pause/CANCEL an active workflow

BEE client

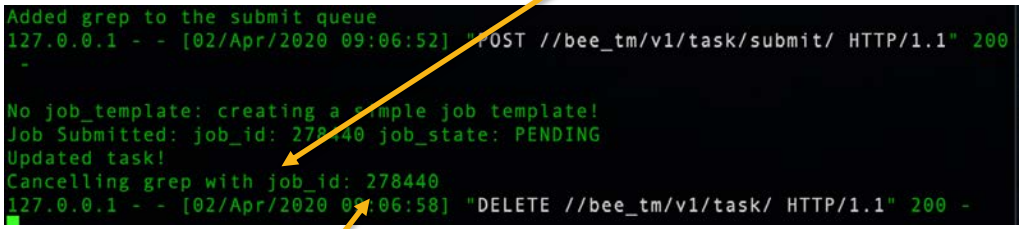
cancel running workflow

BEEWorkflowManager



cancel running task

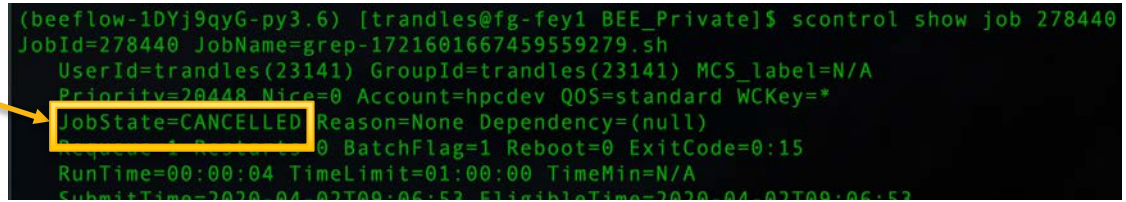
BEETaskManager



When a user cancels a running workflow in BEE, all running jobs are cancelled and workflow execution is halted.

Slurm resource manager

cancel running job



4. Submit ready tasks to the BEETaskManager

5. Report back to the BEE client the status of the submitted workflow and its tasks

```
Job submitted! Your workflow id is 42.
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 client]$ python client.py
Welcome to BEE Client!
0) Submit Workflow
1) Start Workflow
2) Query Workflow
3) Pause Workflow
4) Resume Workflow
5) Cancel Workflow
6) Exit
$ 2
What is the workflow id?
$ 42
STATUS
wc--WAITING
grep--WAITING
```

BEE client

```
What is the workflow id?
$ 42
STATUS
wc--WAITING
grep--PENDING
```

```
What is the workflow id?
$ 42
STATUS
wc--WAITING
grep--RUNNING
```

```
What is the workflow id?
$ 42
STATUS
wc--WAITING
grep--COMPLETED
```

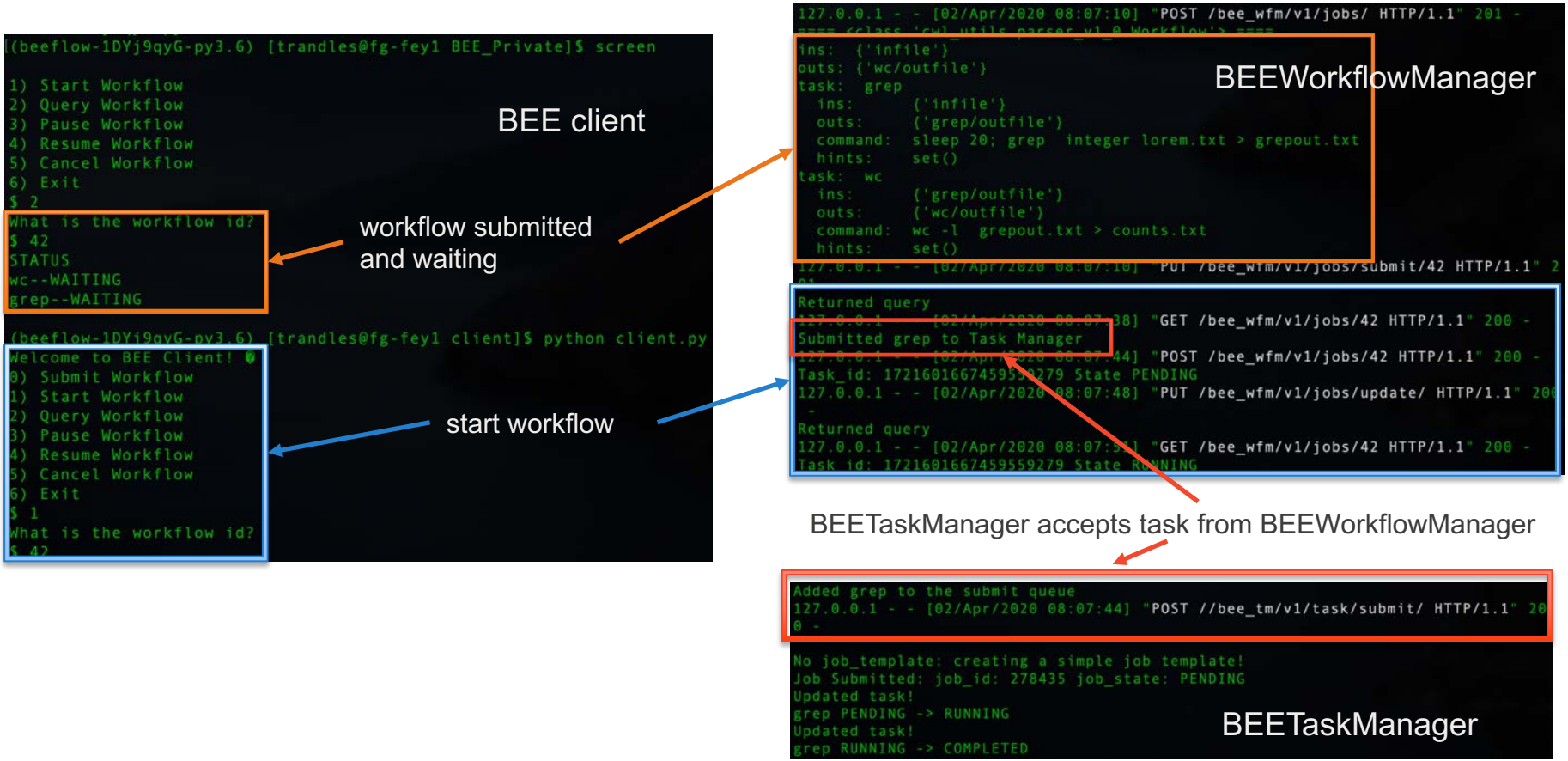
```
What is the workflow id?
$ 42
STATUS
wc--PENDING
grep--COMPLETED
```

For the sake of screen real estate, we only present the relevant pieces of the BEE client output. Each of these correspond to the results of a client workflow query to the BEEWorkflowManager. Refer to slide 18 to see the BEEWorkflowManager and BEETaskManager communication of task state.

```
127.0.0.1 - - [02/Apr/2020 14:22:10] "POST /bee_wfm/v1/jobs/ HTTP/1.1" 201 -
==== <class 'cwl_utils.parser_v1_0.Workflow'> ====
ins:  {'infile'}
outs: {'wc/outfile'}
task:  grep
  ins:      {'infile'}
  outs:     {'grep/outfile'}
  command:  sleep 20; grep  integer lorem.txt > grepout.txt
  hints:    set()
task:  wc
  ins:      {'grep/outfile'}
  outs:     {'wc/outfile'}
  command:  wc -l  grepout.txt > counts.txt
  hints:    set()
127.0.0.1 - - [02/Apr/2020 14:22:10] "PUT /bee_wfm/v1/jobs/submit/42 HTTP/1.1" 201 -
Returned query
127.0.0.1 - - [02/Apr/2020 14:22:16] "GET /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Submitted grep to Task Manager
127.0.0.1 - - [02/Apr/2020 14:22:21] "POST /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 4405064441775592217 State PENDING
127.0.0.1 - - [02/Apr/2020 14:22:23] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200 -
Returned query
127.0.0.1 - - [02/Apr/2020 14:22:28] "GET /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 4405064441775592217 State RUNNING
127.0.0.1 - - [02/Apr/2020 14:22:28] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200 -
Returned query
127.0.0.1 - - [02/Apr/2020 14:22:35] "GET /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 4405064441775592217 State COMPLETED
Submitted wc to Task Manager
127.0.0.1 - - [02/Apr/2020 14:22:48] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200 -
Returned query
127.0.0.1 - - [02/Apr/2020 14:22:50] "GET /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 6441036543068293165 State PENDING
127.0.0.1 - - [02/Apr/2020 14:22:53] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200 -
Returned query
127.0.0.1 - - [02/Apr/2020 14:22:56] "GET /bee_wfm/v1/jobs/42 HTTP/1.1" 200 -
Task_id: 6441036543068293165 State COMPLETED
Workflow Completed!
127.0.0.1 - - [02/Apr/2020 14:22:58] "PUT /bee_wfm/v1/jobs/update/ HTTP/1.1" 200 -
```

BEEWorkflowManager

6. BEETaskManager can accept a task from the BEEWorkflowManager



7. Format the accepted task as a Slurm job script

```
steps:
  grep:
    run:
      class: CommandLineTool
      inputs:
        pattern:
          type: string
          default: "integer"
          inputBinding: {position: 0}
        infile:
          type: File
          default: lorem.txt
          inputBinding: {position: 1}
      outputs:
        outfile: stdout
      stdout: grepout.txt
      baseCommand: grep
      hints:
        DockerRequirement:
          dockerImageId: "/usr/projects/beedev/toss-tiny-3-5.tar"
    in:
      pattern: pattern
      infile: infile
    out: [outfile]
```

This snippet of the CWL workflow on the left shows how a single step (BEE *task*) is specified. Below is the resultant Slurm job script. You can see how the CWL "hints:" item specifies the container image to use in the job. The `ch-run` Charliecloud command line is formed from the "run:" item in the CWL.

```
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 client]$ cat ~/.beeflow/worker/workflow-20200402-10083
8/grep-1721601667459559279.sh
#! /bin/bash
#SBATCH
module load charliecloud
mkdir -p /tmp/$USER
ch-tar2dir /usr/projects/beedev/toss-tiny-3-5.tar /tmp/$USER
ch-run /tmp/$USER/toss-tiny-3-5 -b $PWD -c /mnt/0 -- grep  integer lorem.txt > grepout.txt
rm -rf /tmp/$USER/toss-tiny-3-5
```

8. Use the Charliecloud linux container runtime to execute the task in the Slurm job

This snippet of the CWL workflow shows the use of CWL's Docker container specification, which is adopted for Charliecloud on HPC systems.

```
      baseCommand: grep
      hints:
        DockerRequirement:
          dockerImageId: "/usr/projects/beedev/toss-tiny-3-5.tar"
    id:
```

The BEETaskManager-generated job script is:

- loading the charliecloud environment module
- readying the user-specified container image

```
(beeflow-1EYj9qyG-py3.6) [trandles@fg-fey1 client]$ cat ~/.beeflow/worker/workflow-20200402-10083
8/grep-17216016674595592 79.sh
#! /bin/bash
#SBATCH
module load charliecloud
mkdir -p /tmp/$USER
ch-tar2dir /usr/projects/beedev/toss-tiny-3-5.tar /tmp/$USER
ch-run /tmp/$USER/toss-tiny-3-5 -b $PWD -c /mnt/0 -- grep  integer lorem.txt > grepout.txt
rm -rf /tmp/$USER/toss-tiny-3-5
```

- using the Charliecloud container runtime to execute the task commands

9. Submit the Slurm job to the HPC cluster

```
Added grep to the submit queue
127.0.0.1 - - [02/Apr/2020 08:03:35] "POST //bee_tm/v1/task/submit/ HTTP/1.1" 20
0 -

No job_template: creating a simple job template!
Job Submitted: job_id: 278433 job_state: PENDING
Updated task!
grep PENDING -> RUNNING
Updated task!
grep RUNNING -> COMPLETED
```

BEETaskManager

BEETaskManager creates Slurm job script

```
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 client]$ cat ~/.beeflow/worker/workflow-20200402-10083
8/grep-1721601667459559279.sh
#! /bin/bash
#SBATCH
module load charliecloud
mkdir -p /tmp/$USER
ch-tar2dir /usr/projects/beedev/toss-tiny-3-5.tar /tmp/$USER
ch-run /tmp/$USER/toss-tiny-3-5 -b $PWD -c /mnt/0 -- grep  integer lorem.txt > grepout.txt
rm -rf /tmp/$USER/toss-tiny-3-5
```

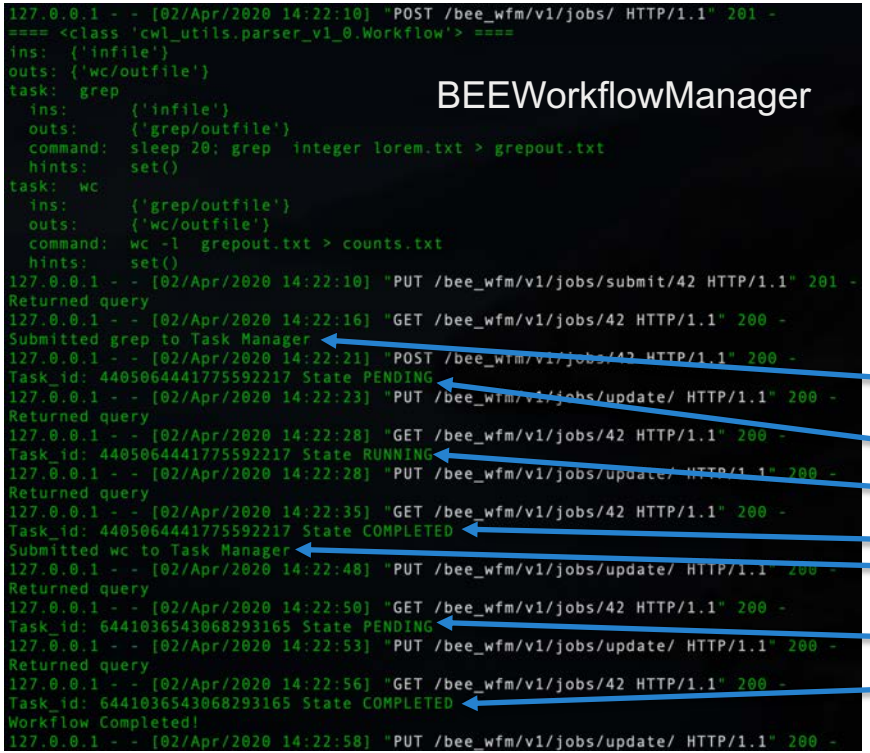
Slurm job script

BEETaskManager submits job to Slurm resource manager

Slurm resource manager

```
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 BEE_Private]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
SON)
(beeflow-1DYj9qyG-py3.6) [trandles@fg-fey1 BEE_Private]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
SON)
            278433  standard grep-172 trandles  R       0:05      1 fg011
```

10. Report back to the BEEWorkflowManager the status of the submitted job



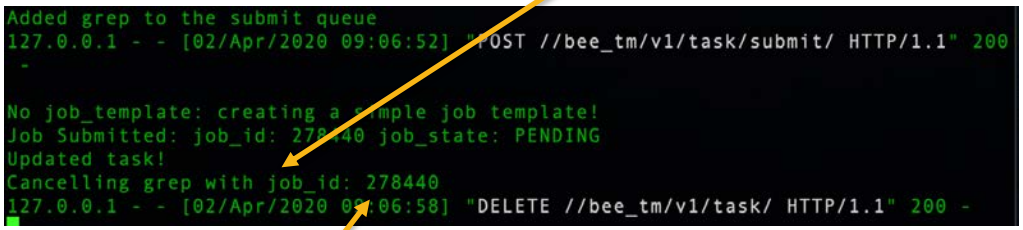
Each of the arrows below shows the task submission and subsequent task state transitions as reported between the BEEWorkflowManager and the BEETaskManager. Refer to slide 13 to see the client display these state transitions as reported by BEEWorkflowManager. Also note that the `wc` workflow task completed so quickly that its state went from `PENDING` to `COMPLETED` faster than the BEETaskManager could report the transition.

11. Cancel a submitted but not yet completed job when commanded to do so by the BEEWorkflowManager

BEEWorkflowManager

cancel running task

BEETaskManager



When a user cancels a running workflow in BEE, all running jobs are cancelled and workflow execution is halted.

Slurm resource manager

cancel running job