Creating BLAST app for Cyverse Version 3

Ken Youens-Clark

Abstract

How I created a BLAST app for Cyverse.

Citation: Ken Youens-Clark Creating BLAST app for Cyverse. **protocols.io**
dx.doi.org/10.17504/protocols.io.g27byhn
Published: 25 Jan 2017

Protocol

Install Cyverse SDK

Step 1.

If you haven't already, install the Cyverse SDK (https://github.com/cyverse/cyverse-sdk) so you have access to "jobs-submit" and such.

Create Github repo

Step 2.

I created https://github.com/hurwitzlab/muscope-blast to hold the code for the Stampede/Cyverse app.

Stampede apps are recommended (http://developer.agaveapi.co/) to have a 'stampede' directory for the code -- perhaps a different dir for each execution system?

I also created https://github.com/hurwitzlab/ohana for the code to build the BLAST dbs and such.

Create the BLAST dbs

Step 3.

I pulled down the HOT from Cyverse Data Store at /iplant/home/scope/data/delong/HOT224-238.

To ensure I had everything, I ran https://github.com/hurwitzlab/ohana/blob/master/scripts/check-md5.pl6 to check against MD5 sums.

I wrote https://github.com/hurwitzlab/ohana/blob/master/scripts/mk-blast.sh to concatenate all contigs/genes/proteins files into one per type, then index with BLAST.

Load Eggnog annotation dbs

Step 4.

The original Eggnog annotations to the predicted genes were delivered in a format that spread one annotation over two lines, so I wrote https://github.com/hurwitzlab/ohana/blob/master/scripts/merge-genes.pl6 to merge them. As there were 15M annotations, I struggled over how to store and retrieve them. I wanted a database like MySQL or Pg, but it's unlikely I could bring up a daemon-based server on stampede, so I chose SQLite. Problem there is I was quite certain it would be too slow to put 15M in one table, so I decided to make a db for each sample (103 of them). The script https://github.com/hurwitzlab/ohana/blob/master/scripts/pyloader.py will load the dbs.

Create entry script

Step 5.

This can be any language or executable, but I tend to write these in bash. I often call mine 'run.sh' (https://github.com/hurwitzlab/muscope-blast/blob/master/stampede/run.sh) and base it off a template (https://github.com/kyclark/metagenomics-book/blob/master/bash/basic.sh) that accepts named arguments. This script will query the input file(s) to the BLAST dbs and then use any resulting hits to predicted genes to query SQLite for annotations which will be placed into an additional file.

Create "test.sh"

Step 6.

The https://github.com/hurwitzlab/muscope-blast/blob/master/stampede/test.sh is for testing that the app is able to be submitted (via 'sbatch') to SLURM and will run.

Create "app.json"

Step 7.

The https://github.com/hurwitzlab/muscope-blast/blob/master/stampede/app.json file describes the app to Agave so it can be registered. It's most important to define the 'inputs' and 'parameters' with argument names that you will reference via environmental variables in the 'template.sh' script.

Create "template.sh"

Step 8.

The https://github.com/hurwitzlab/muscope-blast/blob/master/stampede/template.sh script will take the arguments from Cyverse as environmental variables defined in your 'app.json''s inputs/parameters. I usually just have this pass the arguments to 'run.sh'.

"files-upload" assets to execution system

Step 9.

Use 'files-upload' to put the app/scripts onto the storage (execution?) system. There is an 'upload' target in the Makefile for this. If you change just one file, you can upload that one e.g.

```
$ files-upload -F run.sh kyclark/applications/muscope-blast-0.0.2/stampede
```

Anytime you change your code, test it locally and then "files-upload" the changes into Cyverse.

Register the app with "apps-addupdate"

Step 10.

Run "apps-addupdate -F app.json" to register the app with Agave.

Create "job.json"

Step 11.

Run "jobs-template $APP > job.json" (e.g, just run "make jobs-template") to generate a JSON template for submitting the job via the Agave API.

Submit "job.json"

Step 12.

Use 'jobs-submit -F job.json' (or 'make jobs-submit') to test if it will submit. Use 'jobs-list | head' to see the status. Your job will be the top one and will start off like 'PENDING' and then 'STAGING,' 'QUEUED,' 'RUNNING,' and then 'FAILED' or 'FINISHED.' Using the job ID (e.g., '5905387002803589606-242ac114-0001-007') with 'jobs-status' and 'jobs-history,' you can find out more about what it is doing, possibly where it is failing. When it is done, you can use 'jobs-output' to see what was created.

The results will land in your "$WORK/<userid>/job-<jobid>-<jobname>" directory.

Submit job via DE

Step 13.

Use https://de.cyverse.org/de/ to find the job and submit there.