CLI Example - Parameterize Job Runs

You can use the following Bash script to execute parameterized job runs on the Trifacta® node. This script accepts parameters to identify the CLI package downloaded to the node and then runs the job, whose output includes an identifier for the current date. In this manner, the script can be run on a daily basis on any number of CLI packages.

The CLI package includes:

- `script.cli` - script file
- `datasources.tsv` - file containing a pointer to the storage location of the source data

These values are provided to the script as command-line parameters (`--script` and `--source`).

Based on the specified parameters, this script does the following:

- Launches the job
- Monitors job execution
- Generates error messages if the execution fails
- Generates a success message if the execution succeeds

```
#!/bin/bash

## Example script to run a scheduled job after updating the date.

## Scan the command line arguments for the script file path and for the datasources.tsv file path
for i in "$@"
do
case $i in
    --script=*)
    ScriptParam="${i#*=}"
    ;;

    --source=*)
    SourceParam="${i#*=}"
    ;;

    *)
    # unknown option
    echo ${i} parameter is not recognized. Please provide --script and --source values.
    echo
    exit 0
    ;;
esac
done
if [${ScriptParam} -eq ""]
then
  echo --script param is required.
  echo
  exit 0
fi
if [${SourceParam} -eq ""]
then
  echo --source param is required.
  echo
  exit 0
fi

## Get the current date.
DATE=`date +%m%d`

## Define Job parameters
AppHost=http://localhost:3005
User=user@trifacta.com
Password=password
```

```
## Hard-coded version: Script='/var/log/myDir/script.cli'
Script='${ScriptParam}'
## Hard-coded version: Data='/var/log/myDir/datasources.tsv'
Data='${SourceParam}'
JobType='spark'
OutputFormats='avro'
OutputPath="/data/common/output/$DATE"
extraArgs="--disable_server_certificate_verification"

## uncomment once daily job commences
## Change the job source date
## NEWPATH="hdfs://namenode:port/path/to/$DATE/file"
## echo $NEWPATH > $Data


## Launch job
echo "/opt/trifacta/bin/trifacta_cli.py run_job --script=$Script --data=$Data --host=$AppHost --
job_type=$JobType --user_name=$User --password=$Password --output_formats=$OutputFormats --
output_path=$OutputPath $extraArgs"  >> stdout.txt
/opt/trifacta/bin/trifacta_cli.py run_job --script=$Script --data=$Data --host=$AppHost --job_type=$JobType --
user_name=$User --password=$Password --output_formats=$OutputFormats --output_path=$OutputPath $extraArgs  >>
stdout.txt 2>> stderr.txt


JobLaunched=$?

if [ "$JobLaunched" -eq 1 ]
then
  echo "Failed to launch job. See stderr.txt for details"
  exit 1
fi

## Parse job id from stdout.txt
JobId=$(cat stdout.txt | sed ':a;N;$!ba;s/\n/ /g;s/.*\Job #\([0-9]*\).*/\1/')

JobInfo=''
JobStatus='Pending'


## If jobId exists..
if [ "$JobId" -ge 0 ]
then
  echo "Job with Id " $JobId " launched"

  ## Start loop to monitor job status
  while [ "$JobStatus" != 'Complete' ] && [ "$JobStatus" != 'Failed' ]
  do
    ## Get job status from server
    command="/opt/trifacta/bin/trifacta_cli.py get_job_status --job_id=$JobId --host=$AppHost --
user_name=$User --password=$Password $extraArgs"

    JobStatus=`$command | sed -e 's/^.*: //'`
    if [ "$JobStatus" != 'Complete' ] && [ "$JobStatus" != 'Failed' ]
    then
      echo "Waiting for job to complete..."
      sleep 20
    fi
  done
else
  echo "Failed to launch job. See stderr.txt for details"
  exit 1
fi

if [ "$JobStatus" = 'Complete' ]
then
  echo "Job "$JobId" is complete."
  echo "Output path is $OutputPath"
else
  echo "Job failed to complete. Job status = "$JobStatus
  exit 1
fi
```

You can use the above as a basic template for execution of any type of CLI command.