HTCondor jobs

HTCondor is a job scheduler. You give HTCondor a file containing commands that tell it how to run jobs. HTCondor locates a machine that can run each job within the pool of machines, packages up the job and ships it off to this execute machine. The jobs run, and output is returned to the machine that submitted the jobs.

For HTCondor to run a job, it must be given details such as the names and location of the executable and all needed input files. These details are specified in the submit description file.

A simple example of an executable is a sleep job that waits for 40 seconds and then exits:

```
#!/bin/bash
# file name: sleep.sh

TIMETOWAIT="40"
echo "sleeping for $TIMETOWAIT seconds"
/bin/sleep $TIMETOWAIT
```

To submit this sample job, we need to specify all the details such as the names and location of the executable and all needed input files creating a submit description file where each line has the formcommand

`name = value`

like this:

```
# Unix submit description file
# sleep.sub -- simple sleep job

executable              = sleep.sh
log                     = sleep.log
output                  = outfile.txt
error                   = errors.txt
should_transfer_files   = Yes
when_to_transfer_output = ON_EXIT
queue
```

Submit local jobs

To submit jobs locally, i.e. from CNAF UI, use the following command:

- **`condor_submit`**. The following options are required:
  - **`-name sn-02.cr.cnaf.infn.it:`** to correctly address the job to the submit node;
  - **`-spool:`** to transfer the input files and keep a local copy of the output files;
    - if **`-spool`** is not requested than the user should use **`-remote sn-02.cr.cnaf.infn.it`** instead of **`-name sn-02.cr.cnaf.infn.it`**
  - the submit description file (a `.sub` file containing the relevant information for the batch system), to be indicated as argument.

For example:

```
-bash-4.2$ condor_submit -name sn-02.cr.cnaf.infn.it -spool sleep.sub
Submitting job(s).
1 job(s) submitted to cluster 6798880.
```

where 6798880 is the cluster id.

To see all jobs launched by a user locally on a submit node use

```
condor_q -name sn-02.cr.cnaf.infn.it <user>
```

For example:

```
-bash-4.2$ condor_q -name sn-02.cr.cnaf.infn.it arendina

-- Schedd: sn-02.cr.cnaf.infn.it : <131.154.192.58:9618?... @ 07/29/20 11:11:57
OWNER BATCH_NAME SUBMITTED DONE RUN IDLE TOTAL JOB_IDS
arendina ID: 6785585 7/28 15:35 _ _ _ 1 6785585.0
arendina ID: 6785603 7/28 15:46 _ _ _ 1 6785603.0
arendina ID: 6798880 7/29 10:19 _ _ _ 1 6798880.0
Total for query: 3 jobs; 3 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
Total for all users: 47266 jobs; 35648 completed, 3 removed, 6946 idle, 4643 running, 26 held, 0 suspended
```

To get the list of held jobs and the held reason add the option

`-held`

Whereas, to see information about a single job use

```
condor_q -name sn-02.cr.cnaf.infn.it <cluster id>
```

To investigate why does a job end up in a 'Held' state:

```
condor_q -name sn-02.cr.cnaf.infn.it <cluster id> -af HoldReason
```

and to get more detailed information use the option

`-better-analyze`

For example:

```
-bash-4.2$ condor_q -better-analyze -name sn-02.cr.cnaf.infn.it 6805590

-- Schedd: sn-02.cr.cnaf.infn.it : <131.154.192.58:9618?...
The Requirements expression for job 6805590.000 is

(TARGET.Arch == "X86_64") && (TARGET.OpSys == "LINUX") && (TARGET.Disk >= RequestDisk) &&
(TARGET.Memory >= RequestMemory) && (TARGET.HasFileTransfer)

Job 6805590.000 defines the following attributes:

DiskUsage = 20
ImageSize = 275
MemoryUsage = ((ResidentSetSize + 1023) / 1024)
RequestDisk = DiskUsage
RequestMemory = ifthenelse(MemoryUsage =!= undefined,MemoryUsage,(ImageSize + 1023) / 1024)
ResidentSetSize = 0

The Requirements expression for job 6805590.000 reduces to these conditions:

Slots
Step Matched Condition
----- -------- ---------
[0] 24584 TARGET.Arch == "X86_64"
[1] 24584 TARGET.OpSys == "LINUX"
[3] 24584 TARGET.Disk >= RequestDisk
[5] 24584 TARGET.Memory >= RequestMemory
[7] 24584 TARGET.HasFileTransfer


6805590.000: Job is completed.

Last successful match: Wed Jul 29 16:37:03 2020


6805590.000: Run analysis summary ignoring user priority. Of 829 machines,
0 are rejected by your job's requirements
122 reject your job because of their own requirements
0 match and are already running your jobs
0 match but are serving other users
707 are able to run your job
```

It is possible to format the output of `condor_q` with the option `-af` :

-af list specific attributes
-af:j shows the attribute names
-af:th formats a nice table

The job outputs cannot be copied automatically. The user should launch:

```
condor_transfer_data -name sn-02.cr.cnaf.infn.it <cluster id>
```

with the cluster id returned by `condor_submit` command at submission:

```
-bash-4.2$ condor_transfer_data -name sn-02.cr.cnaf.infn.it 6806037
Fetching data files...
-bash-4.2$ ls
ce_testp308.sub errors.txt outfile.txt sleep.log sleep.sh sleep.sub test.sub
```

At the end, to remove a job use the command **`condor_rm`:**

```
-bash-4.2$ condor_rm -name sn-02.cr.cnaf.infn.it 6806037
All jobs in cluster 6806037 have been marked for removal
```

Also, to fix the submit node you want to submit the job you can launch the command

```
export _condor_SCHEDD_HOST=sn-02.cr.cnaf.infn.it
```

and the commands to submit and check the job become easier:

```
-bash-4.2$ export _condor_SCHEDD_HOST=sn-02.cr.cnaf.infn.it
-bash-4.2$ condor_submit -spool sleep.sub
Submitting job(s).
1 job(s) submitted to cluster 8178760.
-bash-4.2$ condor_q 8178760

-- Schedd: sn-02.cr.cnaf.infn.it : <131.154.192.58:9618?... @ 09/02/20 10:25:30
OWNER BATCH_NAME SUBMITTED DONE RUN IDLE TOTAL JOB_IDS
arendina ID: 8178760 9/2 10:25 _ _ 1 1 8178760.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
Total for all users: 32798 jobs; 13677 completed, 2 removed, 13991 idle, 5115 running, 13 held, 0 suspended
```

Submit grid jobs

With SciTokens

A token can be obtained from a command-line using `oidc-agent`. The oidc agent has to be started:

```
[ashtimmerman@ui-tier1 ~]$ eval `oidc-agent-service use`
```

This starts the agent and sets the required environment variables.

Create an agent account configuration with oidc-gen: (this needs to be done only the first time to create a new agent client)

```
[ashtimmerman@ui-tier1 ~]$ oidc-gen -w device
Enter short name for the account to configure: test_client
[1] https://iam.extreme-datacloud.eu/
[2] https://iam-t1-computing.cloud.cnaf.infn.it/
[3] https://iam-juno.cloud.cnaf.infn.it/
[4] https://iam-belle.cloud.cnaf.infn.it/
[5] https://iam-herd.cloud.cnaf.infn.it/
[6] https://iam.cloud.infn.it/
[7] https://iam-ildg.cloud.cnaf.infn.it/
[8] https://iam-ctao.cloud.cnaf.infn.it/
[9] https://iam-test.indigo-datacloud.eu/
[10] https://iam.deep-hybrid-datacloud.eu/
[11] https://iam-demo.cloud.cnaf.infn.it/
[12] https://b2access.eudat.eu:8443/oauth2
[13] https://b2access-integration.fz-juelich.de/oauth2
[14] https://login-dev.helmholtz.de/oauth2
[15] https://login.helmholtz.de/oauth2
[16] https://services.humanbrainproject.eu/oidc/
[17] https://accounts.google.com
[18] https://aai-dev.egi.eu/auth/realms/egi
[19] https://aai-demo.egi.eu/auth/realms/egi
[20] https://aai.egi.eu/auth/realms/egi
[21] https://login.elixir-czech.org/oidc/
[22] https://oidc.scc.kit.edu/auth/realms/kit
[23] https://wlcg.cloud.cnaf.infn.it/
Issuer [https://iam.extreme-datacloud.eu/]: https://iam-t1-computing.cloud.cnaf.infn.it/
The following scopes are supported: openid profile email address phone offline_access
eduperson_scoped_affiliation eduperson_entitlement eduperson_assurance entitlements
Scopes or 'max' (space separated) [openid profile offline_access]: profile wlcg.groups wlcg compute.create
compute.modify compute.read compute.cancel
Registering Client ...
Generating account configuration ...
accepted
Using a browser on any device, visit: https://iam-t1-computing.cloud.cnaf.infn.it/device
And enter the code: *****
Alternatively you can use the following QR code to visit the above listed URL.
Enter encryption password for account configuration 'test_client': ********
Confirm encryption Password: ********
Everything setup correctly!
```

The `-w device` instructs `oidc-agent` to use the device code flow for the authentication, which is the recommended way with IAM. oidc-agent will display a list of different providers that can be used for registration:

```
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-test.indigo-datacloud.eu/
...
[20] https://oidc.scc.kit.edu/auth/realms/kit/
```

Select one of the registered providers, or type a custom issuer (for IAM, the last character of the issuer string is always a `/`, e.g. `https://wlcg.cloud.cnaf.infn.it/`).

Then `oidc-agent` asks for the scopes, typing `max` (without quotes) allows to get all the allowed scopes, but this is not recommended. Instead specify the minimum required scopes for the task the client is registered for. In the case of job submission to HTCondor-CE, these scopes are:

- `compute.create`
- `compute.modify`
- `compute.read`
- `compute.cancel`

`oidc-agent` will register a new client and store the client credentials.

The client will request to authorize the client to operate on your behalf by requesting to autenticate into IAM with your browser and enter a code provided on the terminal to the indicated web address.

Finally, a password to encrypt the client information on the machine is prompted to the user twice to be set. It can be any password of choice and it must be remembered whenever the client has to be loaded with the `oidc-add <client_name>` command.

After a client is registered, it can be used, in our example with the `test_client` client to obtain access tokens. One does not need to run `oidc-gen` again unless to update or create a new client configuration.

```
[ashtimmerman@ui-tier1 ~]$ oidc-add test_client
Enter decryption password for account config 'test_client':
success
```

Once you've loaded the account, you can use `oidc-token` to get tokens for that account

This command will request the token with oidc-token and export the BEARER_TOKEN environment variable.

```
[ashtimmerman@ui-tier1 ~]$ export BEARER_TOKEN=$(oidc-token test_client)
```

Access tokens are valid typically for 60 minutes.

The scopes determine the actions that can be performed with the token.

To submit a job, save the token in a file by using the following command:

```
[ashtimmerman@ui-tier1 ~]$ mask=$(umask); umask 0077 ; oidc-token test_token > ${HOME}/token ; umask $mask
```

It limits the permissions to the token file to be readable and writeable only by the file owner.

Example of a submit file for job submission with scitokens:

```
[ashtimmerman@ui-tier1 ~]$ cat token_sleep.sub
# Unix submit description file
# sleep.sub -- simple sleep job

scitokens_file = $ENV(HOME)/token
+owner = undefined

executable              = sleep.sh
log                     = sleep.log
output                  = outfile.txt
error                   = errors.txt
should_transfer_files   = Yes
when_to_transfer_output = ON_EXIT
queue
```

Where *`scitokens_file = $ENV(HOME)/token`* is the path to the file containing the scitoken to authenticate to the CE.

`Job submission:`

```
[ashtimmerman@ui-tier1 ~]$ export _condor_SEC_CLIENT_AUTHENTICATION_METHODS=SCITOKENS
[ashtimmerman@ui-tier1 ~]$ condor_submit -pool ce07-htc.cr.cnaf.infn.it:9619 -remote ce07-htc.cr.cnaf.infn.it -
spool token_sleep.sub
Submitting job(s).
1 job(s) submitted to cluster 4037450.

[ashtimmerman@ui-tier1 ~]$ condor_q -pool ce07-htc.cr.cnaf.infn.it:9619 -name ce07-htc.cr.cnaf.infn.it 4037450

 -- Schedd: ce07-htc.cr.cnaf.infn.it : <131.154.192.106:25329?... @ 12/07/23 17:56:19
OWNER          BATCH_NAME     SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
ashtimmermanus ID: 4037450  12/7  14:40      _      _      _      1 4037450.0

Total for query: 1 jobs; 1 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
Total for all users: 8752 jobs; 4843 completed, 0 removed, 591 idle, 3317 running, 1 held, 0 suspended
```

With VOMS proxies

First, create the proxy:

```
voms-proxy-init --voms <vo name>
```

then you can submit the job with the following commands:

```
export _condor_SEC_CLIENT_AUTHENTICATION_METHODS=GSI
condor_submit -pool ce02-htc.cr.cnaf.infn.it:9619 -remote ce02-htc.cr.cnaf.infn.it -spool sleep.sub
```

For example:

```
-bash-4.2$ export _condor_SEC_CLIENT_AUTHENTICATION_METHODS=GSI
-bash-4.2$ condor_submit -pool ce02-htc.cr.cnaf.infn.it:9619 -remote ce02-htc.cr.cnaf.infn.it -spool sleep.sub
Submitting job(s).
1 job(s) submitted to cluster 2015349.
```

where "`sleep.sub`" is the submit file:

```
# Unix submit description file
# sleep.sub -- simple sleep job

use_x509userproxy = true
# needed for all the operation where a certificate is required

+owner = undefined

delegate_job_GSI_credentials_lifetime = 0
# this has to be included if the proxy will last more than 24h, otherwise it will be reduced to 24h
automatically

executable              = sleep.sh
log                     = sleep.log
output                  = outfile.txt
error                   = errors.txt
should_transfer_files   = Yes
when_to_transfer_output = ON_EXIT
queue
```

that differs from the `sleep.sub` file that has been submitted locally by the command name:

`+owner = undefined`

that allows the computing element to identify the user through the voms-proxy.

<u>Note that the submit description file of a grid job is basically different from one that has to be submitted locally.</u>

To check the job status of a single job use

```
condor_q -pool ce02-htc.cr.cnaf.infn.it:9619 -name ce02-htc.cr.cnaf.infn.it <cluster id>
```

So, for the previous example we have:

```
-bash-4.2$ condor_q -pool ce02-htc.cr.cnaf.infn.it:9619 -name ce02-htc.cr.cnaf.infn.it 2015349

-- Schedd: ce02-htc.cr.cnaf.infn.it : <131.154.192.41:9619?... @ 07/29/20 17:02:21
OWNER BATCH_NAME SUBMITTED DONE RUN IDLE TOTAL JOB_IDS
virgo008 ID: 2015349 7/29 16:59 _ _ 1 1 2015349.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
Total for all users: 31430 jobs; 8881 completed, 2 removed, 676 idle, 1705 running, 20166 held, 0 suspended
```

The user is mapped through the voms-proxy in the user name virgo008 as owner of the job. Then, to get the list of the submitted jobs of a user just change **`<cluster id>`** with **`<owner>`**:

```
-bash-4.2$ condor_q -pool ce02-htc.cr.cnaf.infn.it:9619 -name ce02-htc.cr.cnaf.infn.it virgo008

-- Schedd: ce02-htc.cr.cnaf.infn.it : <131.154.192.41:9619?... @ 07/29/20 17:09:42
OWNER BATCH_NAME SUBMITTED DONE RUN IDLE TOTAL JOB_IDS
virgo008 ID: 2014655 7/29 11:30 _ _ _ 1 2014655.0
virgo008 ID: 2014778 7/29 12:24 _ _ _ 1 2014778.0
virgo008 ID: 2014792 7/29 12:40 _ _ _ 1 2014792.0
virgo008 ID: 2015159 7/29 15:11 _ _ _ 1 2015159.0
virgo008 ID: 2015161 7/29 15:12 _ _ _ 1 2015161.0
virgo008 ID: 2015184 7/29 15:24 _ _ _ 1 2015184.0
virgo008 ID: 2015201 7/29 15:33 _ _ _ 1 2015201.0
virgo008 ID: 2015207 7/29 15:39 _ _ _ 1 2015207.0
virgo008 ID: 2015217 7/29 15:43 _ _ _ 1 2015217.0
virgo008 ID: 2015224 7/29 15:47 _ _ _ 1 2015224.0
virgo008 ID: 2015349 7/29 16:59 _ _ _ 1 2015349.0

Total for query: 11 jobs; 11 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
Total for all users: 31429 jobs; 8898 completed, 3 removed, 591 idle, 1737 running, 20200 held, 0 suspended
```

As in the local case, to get the job outputs the user shoud launch:

```
condor_transfer_data -pool ce02-htc.cr.cnaf.infn.it:9619 -name ce02-htc.cr.cnaf.infn.it <cluster id>
```

with the cluster id returned by `condor_submit` command at submission:

```
-bash-4.2$ condor_transfer_data -pool ce02-htc.cr.cnaf.infn.it:9619 -name ce02-htc.cr.cnaf.infn.it 2015217
Fetching data files...
-bash-4.2$ ls
ce_testp308.sub errors.txt outfile.txt sleep.log sleep.sh sleep.sub test.sub
```

And to remove a job submitted via grid:

```
-bash-4.2$ condor_rm -pool ce02-htc.cr.cnaf.infn.it:9619 -name ce02-htc.cr.cnaf.infn.it 2015217
All jobs in cluster 2015217 have been marked for removal
```

Long lived proxy inside a job

One problem which may occurr while running a grid job, is represented by the default short-life of the VOMS proxy used to submit the job. Indeed, the job will be aborted if it does not finish before the expiration time of the proxy. The easiest solution to this problem would be to use very long-lived proxies, but at the expense of an increased security risk. Furthermore, the duration of a VOMS proxy is limited by the VOMS server and cannot be made arbitrarily long.

To overcome this limitation, a proxy credential repository system is used, which allows the user to create and store a long-term proxy in a dedicated server (a "MyProxy" server). At Tier-1 this MyProxy store is myproxy.cnaf.infn.it.

For instance, using the following command, it is possible to retrieve a grid proxy with a lifetime of 168 hours and store it into a the Myproxy server with credentials valid for 720 hours:

```
[dlattanzio@ui-tier1 ~]$ myproxy-init --proxy_lifetime 168 --cred_lifetime 720 --voms vo.padme.org --pshost
myproxy.cnaf.infn.it --dn_as_username --credname proxyCred --local_proxy
Enter GRID pass phrase for this identity:
Contacting voms2.cnaf.infn.it:15020 [/DC=org/DC=terena/DC=tcs/C=IT/ST=Roma/O=Istituto Nazionale di Fisica
Nucleare/OU=CNAF/CN=voms2.cnaf.infn.it] "vo.padme.org"...
Remote VOMS server contacted succesfully.

voms2.cnaf.infn.it:15020: The validity of this VOMS AC in your proxy is shortened to 86400 seconds!

Created proxy in /tmp/myproxy-proxy.10164.21287.

Your proxy is valid until Sat Dec 24 18:34:42 CET 2022
Enter MyProxy pass phrase:
Verifying - Enter MyProxy pass phrase:
Your identity: /DC=org/DC=terena/DC=tcs/C=IT/O=Istituto Nazionale di Fisica Nucleare/CN=Daniele Lattanzio
dlattanzio@infn.it
Creating proxy ............................................ Done
Proxy Verify OK
Your proxy is valid until: Thu Dec  1 18:34:50 2022
A proxy valid for 720 hours (30.0 days) for user /DC=org/DC=terena/DC=tcs/C=IT/O=Istituto Nazionale di Fisica
Nucleare/CN=Daniele Lattanzio dlattanzio@infn.it now exists on myproxy.cnaf.infn.it.
```

As seen above, the user will be requested first to insert the GRID certificate password, and then a "MyProxy pass phrase" for future proxy retrievals.

The same password has to be indicated also in the submission file, which will be similiar to the one in the example below:

```
# Unix submit description file
# sleep2.sub -- simple sleep job

use_x509userproxy = true
# needed for all the operation where a certificate is required

+owner = undefined

MyProxyHost = myproxy.cnaf.infn.it:7512
MyProxyPassword = ***
MyProxyCredentialName = proxyCred
MyProxyRefreshThreshold = 3300
MyProxyNewProxyLifetime = 1440

delegate_job_GSI_credentials_lifetime = 0
# this has to be included if the proxy will last more than 24h, otherwise it will be reduced to 24h
automatically

executable              = sleep.sh
log                     = sleep.log
output                  = outfile.txt
error                   = errors.txt
should_transfer_files   = Yes
when_to_transfer_output = ON_EXIT
queue
```

where `MyProxyRefreshThreshold` and `MyProxyNewProxyLifetime` represent respectively the time (in second) before the expiration of a proxy that the proxy should be refreshed and the new lifetime (in minutes) of the proxy after it is refreshed.

Experiment share usage

If a user wants to know the usage of an entire experiment goup, in particular to see the number of jobs submitted by each experiment user, the command is:

```
condor_q -all -name sn-02 -cons 'AcctGroup == "<exp-name>"' -af Owner jobstatus | sort | uniq -c
```

The output will look like this:

```
-bash-4.2$ condor_q -all -name sn-02 -cons 'AcctGroup == "pulp-fiction"' -af Owner jobstatus | sort | uniq -c
1 MWallace 1
3 VVega 4
20 Wolf 4
572 JulesW 1
1606 Ringo 4
1 Butch 2
5 Jody 4
```

In the first column there is the number of submitted jobs, in the second there is the user who has submitted them and in the third there is the jobs' status (**1 =pending, 2=running, 3=removed, 4=completed, 5=held, 6=submission_err**).