# Installing HTCondor on Ubuntu 18.04

Install HTCondor either from the homepage https://research.cs.wisc.edu/htcondor/instructions/ubuntu/18/stable/ and follow the instructions. Avoid the one on apt by default (it had some weird behavior I can't remember). When finished, run

```
sudo service condor start
```

Check that the pool is active with

```
condor_status
```

## Troubleshooting after install

### Hostname issues

A problem with hostnames can solved by editing /etc/hosts in the way described here: https://papamarkou.blog/2012/01/18/sge-on-single-pc/ , changing

```
127.0.0.1   localhost
127.0.1.1   my_hostname
```

to

```
127.0.0.1   localhost my_hostname
```

This seems to affect the central manager only. New nodes aren't affected.

### Restart properly

If there is any issue so far, you can try stopping the condor service completely and starting again. Do

```
sudo service condor stop
sudo killall condor_master ...
```

where `...` is any other condor service that may be active. Then start again with

```
sudo service condor start
```

Make sure you start condor as a service and not through ´sudo condor_master´, this way it will be easier to manage (start/restart and such).

### Check logs

If anything went wrong you'll know what exactly from checking logs. You'll find these in

```
/var/log/condor/...
```

there is one log file per daemon. Usually `/var/log/condor/MasterLog` and `/var/log/condor/CollectorLog` are the most helpful.

### Use flags like -debug

You'll get more complete reports by using flags like `-debug` when running condor commands. For instance if there are issues with authorization, `condor_status -debug` or `condor_reconfig -debug` will tell you.

# Set up SSL

We will be using SSL to authenticate hosts, users and daemons. The approach used here is taken from these three sources:

[http://pages.cs.wisc.edu/~zmiller/ca-howto/](http://pages.cs.wisc.edu/~zmiller/ca-howto/)

[https://stackoverflow.com/questions/21297139/how-do-you-sign-a-certificate-signing-request-with-your-certification-authority/21340898#21340898](https://stackoverflow.com/questions/21297139/how-do-you-sign-a-certificate-signing-request-with-your-certification-authority/21340898#21340898)

and

[https://www.akadia.com/services/ssh_test_certificate.html](https://www.akadia.com/services/ssh_test_certificate.html)

The steps to generate SSL certificates are all done on the host machine, and node certificates are distributed later to the node machines.

1. Set up a certificate signing authority, essentially files `cacert.key` and `cacert.crt`. Use the config file `openssl-ca.cnf` to do this.
2. Use the signing authority to generate a host certificate, `host.key` and `host.crt`. Keep this one in any folder on your host machine. Use the config file `openssl-host.cnf` to do this.
3. Use the signing authority to generate different node certificates which can be distributed to the nodes in the cluster. Use the config file `openssl-node.cnf` to do this.

## 1. Set up the certificate signing authority key pair

Go to an appropriate starting directory and start by creating the configuration file, `touch openssl-ca.cnf` and add the following:

```
HOME            = .
RANDFILE        = $ENV::HOME/.rnd

####################################################################
[ ca ]
default_ca    = CA_default      # The default ca section

[ CA_default ]
#Second, add the following to the [ CA_default ] section of openssl-ca.cnf. I left them
out earlier because th$
base_dir      = .
certificate   = $base_dir/cacert.crt   # The CA certifcate
private_key   = $base_dir/cacert.key    # The CA private key
new_certs_dir = $base_dir              # Location for new certs after signing
database      = $base_dir/index.txt    # Database index file
serial        = $base_dir/serial.txt   # The current serial number

unique_subject = yes # Set to 'no' to allow creation of
                     # several certificates with same subject.

default_days     = 1000         # how long to certify for
default_crl_days = 30           # how long before next CRL
default_md       = sha256       # use public key default MD
preserve         = no           # keep passed DN ordering

x509_extensions = ca_extensions # The extensions to add to the cert

email_in_dn     = no            # Don't concat the email in the DN
copy_extensions = copy          # Required to copy SANs from CSR to cert

####################################################################



[ req ]
default_bits       = 4096
default_keyfile    = cacert.key
distinguished_name = ca_distinguished_name
x509_extensions    = ca_extensions
string_mask        = utf8only

####################################################################
[ ca_distinguished_name ]
countryName         = Country Name (2 letter code)
countryName_default = SE

stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = Stockholm

localityName                = Locality Name (eg, city)
localityName_default        = Stockholm


organizationName            = Organization Name (eg, company)
```

```
organizationName_default    = Royal Institute of Technology -- KTH

organizationalUnitName         = Organizational Unit (eg, division)
organizationalUnitName_default = Condensed Matter Physics

commonName         = Common Name (e.g. server FQDN or YOUR name)
commonName_default = ROOT CA

emailAddress         = Email Address
#emailAddress_default = test@example.com

####################################################################
[ ca_extensions ]

subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always, issuer
basicConstraints       = critical, CA:true
keyUsage               = keyCertSign, cRLSign

####################################################################
[ signing_policy ]
countryName            = optional
stateOrProvinceName    = optional
localityName           = optional
organizationName       = optional
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional

####################################################################
[ signing_req ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints       = CA:FALSE
keyUsage               = digitalSignature, keyEncipherment
```

Now create the signing certificates (might need sudo)

```
openssl req -x509 -config openssl-ca.cnf -newkey rsa:4096 -sha256 -out cacert.crt
```

Follow the steps on screen, use all the defaults (do not give an email for some reason...).

This creates two files `cacert.crt` and `cacert.key`, where the latter is encrypted with the password you provided. To avoid password encryption, use the flag `-nodes` (bad idea!).

## 2. Generate the host certificate key pair

Create the file `touch openssl-host.cnf` and add

```
HOME            = .
RANDFILE        = $ENV::HOME/.rnd

####################################################################
[ req ]
default_bits       = 4096
default_keyfile    = server.key
distinguished_name = server_distinguished_name
req_extensions     = server_req_extensions
string_mask        = utf8only

####################################################################
[ server_distinguished_name ]
countryName         = Country Name (2 letter code)
countryName_default = SE

stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = Stockholm

localityName         = Locality Name (eg, city)
localityName_default = Stockholm

organizationName            = Organization Name (eg, company)
organizationName_default    = Royal Institute of Technology -- KTH

commonName           = Common Name (e.g. server FQDN or YOUR name)
commonName_default   = ServerName

emailAddress         = Email Address
#emailAddress_default = test@example.com

####################################################################
[ server_req_extensions ]

subjectKeyIdentifier = hash
basicConstraints     = CA:FALSE
keyUsage             = digitalSignature, keyEncipherment
subjectAltName       = @alternate_names
nsComment            = "OpenSSL Generated Certificate"

####################################################################
[ alternate_names ]

DNS.1  = example.com
DNS.2  = www.example.com
DNS.3  = mail.example.com
DNS.4  = ftp.example.com
```

Now create a certificate request to be signed by the authority (a "csr"), and a private key in one go (might need sudo)

```
openssl req -config openssl-server.cnf -newkey rsa:4096 -sha256 -nodes -out server.csr
```

Once again follow the instructions on screen. To avoid password encryption use `-nodes` above as well.

Now you have the certificate request `host.csr` and encrypted key `host.key`.

Next, we sign this request with our certificate authority (CA). Start by creating these files

```
touch index.txt
echo '01' > serial.txt
```

Then sign,

```
openssl ca -config openssl-ca.cnf -policy signing_policy -extensions signing_req -out
server.crt -infiles server.csr
```

If you receive weird errors, retry with sudo, or check that you are the owner of the files in use.

Now you should have a signed certificate `host.crt`

## 3. Generate node certificates

Create the file `touch openssl-node.cnf` and add

```
HOME            = .
RANDFILE        = $ENV::HOME/.rnd

####################################################################
[ req ]
default_bits       = 4096
default_keyfile    = node.key
distinguished_name = node_distinguished_name
req_extensions     = node_req_extensions
string_mask        = utf8only

####################################################################
[ node_distinguished_name ]
countryName         = Country Name (2 letter code)
countryName_default = SE

stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = Stockholm

localityName         = Locality Name (eg, city)
localityName_default = Stockholm

organizationName            = Organization Name (eg, company)
organizationName_default    = Royal Institute of Technology -- KTH

commonName           = Common Name (e.g. server FQDN or YOUR name)
commonName_default   = nodeName

emailAddress         = Email Address
#emailAddress_default = test@example.com

####################################################################
[ node_req_extensions ]

subjectKeyIdentifier = hash
basicConstraints     = CA:FALSE
keyUsage             = digitalSignature, keyEncipherment
subjectAltName       = @alternate_names
nsComment            = "OpenSSL Generated Certificate"

####################################################################
[ alternate_names ]

DNS.1  = example.com
DNS.2  = www.example.com
DNS.3  = mail.example.com
DNS.4  = ftp.example.com
```

Now the procedure is similar to the issuing the host certificate. In short:

- Create the certificate request (.csr) and private key (.key). Fill in the appropriate fields for your node.

```
openssl req -config openssl-node.cnf -newkey rsa:4096 -sha256 -nodes -out node.csr
```

- Sign the request using the CA

```
openssl ca -config openssl-ca.cnf -policy signing_policy -extensions signing_req -
out node.crt -infiles node.csr
```

- Send the files `cacert.crt`, `node.crt`, `node.key`, to the node so that it may authenticate with your host.

## Troubleshoot SSL

### Remove password encryption from a private key

For some reason condor doesn't like the host and node keys to be encrypted. Of course, the CA key `cacert.key` can remain encrypted (use it to make sure only you can issue new certificates). Entering the "PEM" password for encrypted server/node keys results in authentication failure. Perhaps I've done something wrong in the configuration. In any case, to remove the password encryption from a key file do

```
openssl rsa -in myfile.encrypted.key -out myfile.unencrypted.key
```

# Configure HTCondor

Now that the SSL certificates are issued it's time to configure HTCondor use them.

## Condor Certificate Map File

The map file maps ssl certificates to Condor user names. I set it as a variable in `/etc/condor/condor_config.local` like this on hosts and nodes alke:

```
CERTIFICATE_MAPFILE      = /etc/condor/ssl/condor_map_file
```

The contents look like this:

```
SSL "/C=SE/ST=Stockholm/L=Stockholm/O=Royal Institute of Technology --
KTH/CN=david@thinkstation" david
SSL "/C=SE/ST=Stockholm/L=Stockholm/O=Royal Institute of Technology --
KTH/CN=david@uburku"       david
```

On each line there are three fields, SSL, subject and username.

You can find the subject line in any certificate file. Easiest is to do:

```
openssl x509 -in mycertificate.crt -subject -noout
```

Just make sure you replace the field separator ", " by "/". Each field starts with "/".

# Configure a new host (Central Manager)

You'll find the files that you need to configure in `/etc/condor/...`

To configure the Host, you'll need to edit `/etc/condor/condor_config.local` and create these new files

- `/etc/condor/config.d/40host.config`
- `/etc/condor/config.d/41shared_port.config`
- `/etc/condor/config.d/42node_list.config`
- (optional) `/etc/condor/config.d/43policy.config` (See policy section)

Start by pasting the following into `/etc/condor/condor_config.local`

```
# HTCondor configuration file


#
# Configuration placed into this file extends/overwrites the settings in the
# main HTCondor configuration at /etc/condor/condor_config.
# It may be advantagous to leave the main configuration file pristine and put
# local configuration here to ease configuration updates during upgrades of the
# HTCondor Debian package. Alternatively, it is also possible to place additional
# configuration files into /etc/condor/config.d that will take precedence over
# both the main configuration file and this local configuration. Note that
# DebConf-generated configuration will overwrite settings in this file.

#use ROLE : Personal, CentralManager, Submit
#use SECURITY : USER_BASED

SEC_DEFAULT_AUTHENTICATION         = REQUIRED
SEC_DEFAULT_ENCRYPTION             = REQUIRED
SEC_DEFAULT_INTEGRITY              = REQUIRED

SEC_DAEMON_AUTHENTICATION          = REQUIRED
SEC_DAEMON_ENCRYPTION              = REQUIRED
SEC_DAEMON_INTEGRITY               = REQUIRED

SEC_NEGOTIATOR_AUTHENTICATION      = REQUIRED
SEC_NEGOTIATOR_ENCRYPTION          = REQUIRED
SEC_NEGOTIATOR_INTEGRITY           = REQUIRED

SEC_CLIENT_AUTHENTICATION          = REQUIRED
SEC_CLIENT_ENCRYPTION              = REQUIRED
SEC_CLIENT_INTEGRITY               = REQUIRED

SEC_DEFAULT_AUTHENTICATION_METHODS    = SSL
SEC_DAEMON_AUTHENTICATION_METHODS     = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_NEGOTIATOR_AUTHENTICATION_METHODS = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_READ_AUTHENTICATION_METHODS       = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_WRITE_AUTHENTICATION_METHODS      = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_CLIENT_AUTHENTICATION_METHODS     = $(SEC_DEFAULT_AUTHENTICATION_METHODS)

AUTH_SSL_CLIENT_CAFILE   = /etc/condor/ssl/cacert.crt
AUTH_SSL_CLIENT_CERTFILE = /etc/condor/ssl/host.crt
AUTH_SSL_CLIENT_KEYFILE  = /etc/condor/ssl/host.key
#Make sure this key is not encrypted!

AUTH_SSL_SERVER_CAFILE   = /etc/condor/ssl/cacert.crt
AUTH_SSL_SERVER_CERTFILE = /etc/condor/ssl/host.crt
AUTH_SSL_SERVER_KEYFILE  = /etc/condor/ssl/host.key
#Make sure this key is not encrypted!


CERTIFICATE_MAPFILE      = /etc/condor/ssl/condor_map_file
#NEGOTIATOR_DEBUG= D_FULLDEBUG

#COLLECTOR_DEBUG = D_FULLDEBUG
```

```
#CREDD_DEBUG      = D_FULLDEBUG
#MASTER_DEBUG     = D_FULLDEBUG
```

This takes care of authentication on the host side. Notice that any daemon can be client/server depending on if it initiates/responds to a request. This has nothing to do with being a host or a node. Make sure the SSL certificate files point correctly, and that the key file is not encrypted.

Create `/etc/condor/config.d/40host.config` and paste the following:

```
DAEMON_LIST          = COLLECTOR, NEGOTIATOR, SCHEDD, STARTD, MASTER, CREDD
CONDOR_HOST          = localhost
CREDD_HOST           = $(CONDOR_HOST)
```

This enables the daemons that are supposed to exist on a host.

Create `/etc/condor/config.d/41shared_port.config` and paste the following:

```
SHARED_PORT_ARGS = -p 9618
DAEMON_LIST = $(DAEMON_LIST), SHARED_PORT
COLLECTOR_HOST = $(CONDOR_HOST)
USE_SHARED_PORT = TRUE
UPDATE_COLLECTOR_WITH_TCP = TRUE
TRUST_UID_DOMAIN = TRUE
TCP_FORWARDING_HOST = my.external.ip.address
BIND_ALL_INTERFACES=TRUE
```

This makes connections simpler. All daemons connect through one port, 9618, so this is the only port that needs to be opened in a router for port-forwarding, say. The other ones are nice tricks you can read about in the manual. They all help with connectivity in some way.

Create `/etc/condor/config.d/42host.config` and paste the following:

```
ALLOW_OWNER          = david@localhost, david@thinkstation
ALLOW_ADMINISTRATOR = david@localhost, david@thinkstation
ALLOW_CONFIG         = david@localhost, david@thinkstation
ALLOW_DAEMON         = david@localhost, david@thinkstation,\
                       david@uburku, david@thinkpad, davidace@rosenrot
ALLOW_WRITE          = david@localhost, david@thinkstation,\
                       david@uburku, david@thinkpad, davidace@rosenrot
ALLOW_READ           = david@localhost, david@thinkstation,\
                       david@uburku, david@thinkpad, davidace@rosenrot
```

Here the user names are simply the same ones used in `/etc/condor/ssl/condor_map_file`, and in this case `thinkstation` is the name of the host machine.

## Configure a new node

You'll find the files that you need to configure in `/etc/condor/...`

To configure a node, you'll need to edit `/etc/condor/condor_config.local` and create these new files

- `/etc/condor/config.d/40node.config`
- `/etc/condor/config.d/41shared_port.config` (same as for the host)
- (optional) `/etc/condor/config.d/43policy.config` (See Desktop policy section)

Start by pasting the following into `/etc/condor/condor_config.local`

```
# HTCondor configuration file
#
# Configuration placed into this file extends/overwrites the settings in the
# main HTCondor configuration at /etc/condor/condor_config.
# It may be advantagous to leave the main configuration file pristine and put
# local configuration here to ease configuration updates during upgrades of the
# HTCondor Debian package. Alternatively, it is also possible to place additional
# configuration files into /etc/condor/config.d that will take precedence over
# both the main configuration file and this local configuration. Note that
# DebConf-generated configuration will overwrite settings in this file.


SEC_DEFAULT_AUTHENTICATION         = REQUIRED
SEC_DEFAULT_ENCRYPTION             = REQUIRED
SEC_DEFAULT_INTEGRITY              = REQUIRED

SEC_DAEMON_AUTHENTICATION          = REQUIRED
SEC_DAEMON_ENCRYPTION              = REQUIRED
SEC_DAEMON_INTEGRITY               = REQUIRED

SEC_NEGOTIATOR_AUTHENTICATION      = REQUIRED
SEC_NEGOTIATOR_ENCRYPTION          = REQUIRED
SEC_NEGOTIATOR_INTEGRITY           = REQUIRED

SEC_CLIENT_AUTHENTICATION          = REQUIRED
SEC_CLIENT_ENCRYPTION              = REQUIRED
SEC_CLIENT_INTEGRITY               = REQUIRED

SEC_DEFAULT_AUTHENTICATION_METHODS    = SSL
SEC_DAEMON_AUTHENTICATION_METHODS     = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_NEGOTIATOR_AUTHENTICATION_METHODS = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_READ_AUTHENTICATION_METHODS       = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_WRITE_AUTHENTICATION_METHODS      = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SEC_CLIENT_AUTHENTICATION_METHODS     = $(SEC_DEFAULT_AUTHENTICATION_METHODS)


AUTH_SSL_CLIENT_CAFILE   = /etc/condor/ssl/cacert.crt
AUTH_SSL_CLIENT_CERTFILE = /etc/condor/ssl/node.crt
AUTH_SSL_CLIENT_KEYFILE  = /etc/condor/ssl/node.key
#Make sure this key is not encrypted!

AUTH_SSL_SERVER_CAFILE   = /etc/condor/ssl/cacert.crt
AUTH_SSL_SERVER_CERTFILE = /etc/condor/ssl/node.crt
AUTH_SSL_SERVER_KEYFILE  = /etc/condor/ssl/node.key
#Make sure this key is not encrypted!


CERTIFICATE_MAPFILE      = /etc/condor/ssl/condor_map_file

#MASTER_DEBUG    = D_SECURITY
#SCHEDD_DEBUG    = D_FULLDEBUG
```

Paste the following into `/etc/condor/config.d/40node.config` and edit accordingly.

```
CONDOR_HOST          = host.ip.address
IP_ADDRESS           = my.external.ip.address
DAEMON_LIST          = MASTER, STARTD, SCHEDD
CREDD_HOST           = $(CONDOR_HOST)
COLLECTOR_HOST       = $(CONDOR_HOST)

ALLOW_WRITE          = david@thinkstation, david@uburku
ALLOW_READ           = david@thinkstation, david@uburku
ALLOW_ADMINISTRATOR = david@thinkstation, david@uburku
ALLOW_DAEMON         = david@thinkstation, david@uburku
ALLOW_NEGOTIATOR     = david@thinkstation, david@uburku
ALLOW_CONFIG         = david@thinkstation, david@uburku

IsDesktop = True
```

The `MASTER`, `STARTD` and `SCHEDD` are daemons that make this a condor executing and submitting machine respectively. The "IsDesktop" variable is used to tell condor if this is a desktop or a dedicated machine. The variable is read by the file `/etc/condor/config.d/43policy.config`.

Once again the user names are the ones in condor map file.

Next, paste the following into a new file `/etc/condor/config.d/41shared_port.config`

```
SHARED_PORT_ARGS = -p 9618
DAEMON_LIST = $(DAEMON_LIST), SHARED_PORT
COLLECTOR_HOST = $(CONDOR_HOST)
USE_SHARED_PORT = TRUE
UPDATE_COLLECTOR_WITH_TCP = TRUE
TRUST_UID_DOMAIN = TRUE
TCP_FORWARDING_HOST = my.external.ip.address
BIND_ALL_INTERFACES=TRUE
```

This is identical to the one on the host.

And that's it.

## Desktop policy and priority

Create the file `/etc/condor/config.d/43policy.config` and paste

```
# If "IsDesktop" is configured, make it an attribute of the machine ClassAd.
STARTD_ATTRS = IsDesktop

# Only consider starting jobs if:
# 1) the load average is low enough OR the machine is currently
#    running an HTCondor job
# 2) AND the user is not active (if a desktop)
START = ( ($(CPUIdle) || (State != "Unclaimed" && State != "Owner")) \
          && (IsDesktop =!= True || (KeyboardIdle > $(StartIdleTime))) )

# Suspend (instead of vacating/killing) for the following cases:
WANT_SUSPEND = ( $(SmallJob) || $(JustCpu) \
                 || $(IsVanilla) )

# When preempting, vacate (instead of killing) in the following cases:
WANT_VACATE  = ( $(ActivationTimer) > 10 * $(MINUTE) \
                 || $(IsVanilla) )

# Suspend jobs if:
# 1) The CPU has been busy for more than 2 minutes, AND
# 2) the job has been running for more than 90 seconds
# 3) OR suspend if this is a desktop and the user is active
SUSPEND = ( ((CpuBusyTime > 2 * $(MINUTE)) && ($(ActivationTimer) > 90)) \
            || ( IsDesktop =?= True && $(KeyboardBusy) ) )

# Continue jobs if:
# 1) the CPU is idle, AND
# 2) we've been suspended more than 5 minutes AND
# 3) the keyboard has been idle for long enough (if this is a desktop)
CONTINUE = ( $(CPUIdle) && ($(ActivityTimer) > 300) \
             && (IsDesktop =!= True || (KeyboardIdle > $(ContinueIdleTime))) )

# Preempt jobs if:
# 1) The job is suspended and has been suspended longer than we want
# 2) OR, we don't want to suspend this job, but the conditions to
#    suspend jobs have been met (someone is using the machine)
PREEMPT = ( ((Activity == "Suspended") && \
            ($(ActivityTimer) > $(MaxSuspendTime))) \
           || (SUSPEND && (WANT_SUSPEND == False)) )

# Replace 0 in the following expression with whatever amount of
# retirement time you want dedicated machines to provide.  The other part
# of the expression forces the whole expression to 0 on desktop
# machines.
MAXJOBRETIREMENTTIME = (IsDesktop =!= True) * 5

# Kill jobs if they have taken too long to vacate gracefully
MachineMaxVacateTime = 10 * $(MINUTE)
KILL = False
```

If you've set `IsDesktop = True` somewhere, this will make Condor prioritize the owner of the machine before running jobs on it.

# Submitting jobs

Read this very nice tutorial:

http://www.iac.es/sieinvens/siepedia/pmwiki.php?n=HOWTOs.CondorSubmitFile#creating_submit_files

Here's an example submit file

```
ID          = $(Process)
NAME        = test
universe    = vanilla
executable = ../cmake-build-debug/DMRG++
arguments   = $(inputfiles)
should_transfer_files = yes
when_to_transfer_output = on_exit_or_evict
transfer_input_files = $(inputfiles)
transfer_output_files = output/
transfer_output_remaps = "output.h5 = ../output/output_$(ID).h5"

log         = logs/$(NAME)_$(ID).log
output      = logs/$(NAME)_$(ID).out
error       = logs/$(NAME)_$(ID).error
queue inputfiles matching files ../input/$(NAME)_*.cfg
```

This submit file will run the executable with arguments `..input/test_1.cfg`, `..input/test_2.cfg`, `..input/test_3.cfg`... however many matchin files it finds in the folder `..input/`

It will transfer the input test files to the execute node, and transfer the output file on the execute machine `output/output.h5` back into the folder `../output`/output_#.h5 on the submit machine, where # is the ID of the job.

# Troubleshooting

## Jobs never start when submitting from nodes

There can be many reasons for this error. Check the log `/var/log/condor/SchedLog` to debug. For more debug info use the variable `SCHEDD_DEBUG = D_FULLDEBUG` or even `D_ALL` on both the host and node. Another important log is `var/log/condor/NegotiatorLog` on the host.

For me this error has been fixed by opening port 9618 on the router and/or firewall. For intsance if `ufw` is enabled, do

```
sudo ufw allow 9618
```

## Jobs terminate with errors on other nodes

You have to make sure that your binary is actually compatible with the architecture of the node, and if you use any shared libraries, these have to be present in the nodes as well. To remedy these issues

- Compile statically. Link your libraries statically.
- Compile with generic architecture, for instance with c++ compiler flags such as

  ```
  -march=sandybridge -mtune=native
  ```

## I want to disable hyperthreading

Create the file `/etc/condor/config.d/ht.conf` containing the following line:

```
COUNT_HYPERTHREAD_CPUS=FALSE
```

Activate the setting by restarting condor: `sudo service condor restart`