

3rd Meeting report

1. HTCondor

Overview



Keywords definition in HTCondor

Job: Representation of a piece of work ex. Unix process or element of a workflow

ClassAd: internal data representation

Machine or Resource: Computers that do the processing

Pool: Group of machine/resource

Matchmaking: Assign a job to machine or resource

Central manager: One of pool (machine) that do the matchmaking

Submit Host: Computer from which jobs are submitted to HTCondor

Execute Host: The computer that runs a job

Requirements: ex. Linux x86-64

Rank: Require most memory or cpu

Part of a Job ClassAd

```
MyType      = "Job"           String
TargetType  = "Machine"
ClusterId   = 1               Integer
ProcID      = 0
IsPhysics   = True            Boolean
Owner       = "einstein"
Cmd         = "cosmos"
Requirements = {Arch == "INTEL"} Boolean Expression
```

Example of ClassAd (Requirements for each jobs ex. Architecture/CMD)

Job

Input/output participation

Currently HTCondor doesn't support standard input/output such as keyboard so we use command line arguments and file transfer instead.

Command Line Arguments

```
Universe     = vanilla
Executable   = cosmos
Arguments    = -c 299792458 -G 6.673e-112
. . .
Queue
```

Invokes executable with
`cosmos -c 299792458 -G 6.673e-112`

Look at the `condor_submit` man page to see syntax for Arguments. This example has `argc = 5`.

File Transfer Example

```
# changed cosmos.sub file
Universe     = vanilla
Executable    = cosmos
Log           = cosmos.log
Transfer_Input_Files   = cosmos.dat
Transfer_Output_Files  = results.dat
Should_Transfer_Files  = IF_NEEDED
When_To_Transfer_Output = ON_EXIT
Queue
```

Example job file that configure input/output via command line args and file transfer

Creating job

In brief, we need to create job definition file by ourself before submit to HTCondor

1 Cluster, 2 Jobs

```
Universe     = vanilla
Executable    = cosmos

Log           = cosmos_0.log
Input         = cosmos_0.in
Output        = cosmos_0.out
Queue
job 102.0

Log           = cosmos_1.log
Input         = cosmos_1.in
Output        = cosmos_1.out
Queue
job 102.1
```

Better Organization

- Create a subdirectory for each job, intentionally named
`run_0, run_1, ... run_999999`
- Implement the creation of directories with a program (such as Python or Perl)
- Create or place input files in each of these
`run_0/cosmos.in`
`run_1/cosmos.in`
...
`run_999999/cosmos.in`
- The output and log files for each job will be created by the job, when the job runs.



Example of job definitions and organization

HTCondor Basic Commands

Before we going get started we need to understand the fundamental of HTCondor such as how it works, how to send the job, et cetera. So I decided to following the tutorial that make we understand and clear how it's work.

The executable

Unix executable, a shell script

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

TIMETOWAIT="6"
echo "sleeping for $TIMETOWAIT seconds"
/bin/sleep $TIMETOWAIT
```

The job script content name 'sleep.sh'

For demonstrate the HTCondor we'll create a simply script that wait for 6 seconds and exit and picture above is the executable(script) file name 'sleep.sh'. For a Unix submit machine only, need to add x permission to run file by running Unix command
\$ chmod u+x sleep.sh

The contents of the submit description file

```
# 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
```

The submit description file named 'sleep.sub'

The submit description file describes the job. Comment begins with '#' character, comments do not span lines. Each line of the submit description file has the form with 'command = value' every command is case sensitive.

command name = value

The description file form.

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

executable      = sleep.sh      ; the executable file
log             = sleep.log      ; the log file name
output          = outfile.txt    ; the output file name
error           = errors.txt     ; the error file name
should_transfer_files = Yes
when_to_transfer_output = ON_EXIT
queue           ; tells HTCondor to run one instance of this job.
```

Describes a file command

should_transfer_files = Yes

when_to_transfer_output = ON_EXIT

“According to HTCondor command it’s said

direct HTCondor to explicitly send the needed files, including the executable, to the machine where the job executes. These commands will likely not be necessary for jobs in which the submit machine and the execute machine access a shared file system. However, including these commands will allow this first sample job to work under a large variety of pool configurations.”

Submitting the job

Hence we have the submit description file(‘sleep.sub’) with in current working directory named ‘sleep.sub’ and the executable file(in this case name ‘sleep.sh’), this job submission is accomplished with the command line

```
$ condor_submit sleep.sub
```

If the submission completed, the terminal will display a response that identifies the job, of the form

```
Submitting job(s).
1 job(s) submitted to cluster 6.
```

The output from condor_submit

Monitoring the job

Once the job has been submitted, all we need to know to help us with monitoring a job by using condor_q command for example after we submitted a job and no other jobs in queue the below is the output.

```
% condor_q
-- Submitter: example.wisc.edu : <128.105.14.44:56550> : example.wisc.edu
ID      OWNER      SUBMITTED      RUN_TIME ST PRI SIZE CMD
  6.0    kris        2/13 10:49      0+00:00:03 R  0   97.7 sleep.sh

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```

If the queue contain many jobs and we want to specific a person to view, we'll add the option `-submitter <owner>` in this case we want to view a job that owner by 'kris'.

```
$ condor_q -submitter kris
```

According to "HTCondor Quick Start Guide" is document that describe the condor_q output.

The first column of output from `condor_q` identifies the job; the identifier is composed of two integers separated by a period. The first integer is known as a **cluster number**, and it will be the same for each of the potentially many jobs submitted by this single invocation of `condor_submit`. The second integer in the identifier is known as a **process ID**, and it distinguishes between distinct job instances that have the same cluster number. These values start at 0.

```
000 (006.000.000) 02/13 10:49:04 Job submitted from host: <128.105.14.44:46062>
...
001 (006.000.000) 02/13 10:49:24 Job executing on host: <128.105.15.5:43051?PrivNet=cs.wisc.edu>
...
006 (006.000.000) 02/13 10:49:30 Image size of job updated: 100000
    0 - MemoryUsage of job (MB)
    0 - ResidentSetSize of job (KB)
...
005 (006.000.000) 02/13 10:49:31 Job terminated.
    (1) Normal termination (return value 0)
        Usr 0 00:00:00, Sys 0 00:00:00 - Run Remote Usage
        Usr 0 00:00:00, Sys 0 00:00:00 - Run Local Usage
        Usr 0 00:00:00, Sys 0 00:00:00 - Total Remote Usage
        Usr 0 00:00:00, Sys 0 00:00:00 - Total Local Usage
    23 - Run Bytes Sent By Job
    113 - Run Bytes Received By Job
    23 - Total Bytes Sent By Job
    113 - Total Bytes Received By Job
    Partitionable Resources :      Usage  Request  Allocated
        Cpus                :           1           1
        Disk (KB)           :    100000    100000    2033496
        Memory (MB)         :           0          98       2001
...
```

A content of job completion log file

Removing a job

The `condor_rm` with specific the job identifier to remove a job.

```
$ condor_rm 6.0
```

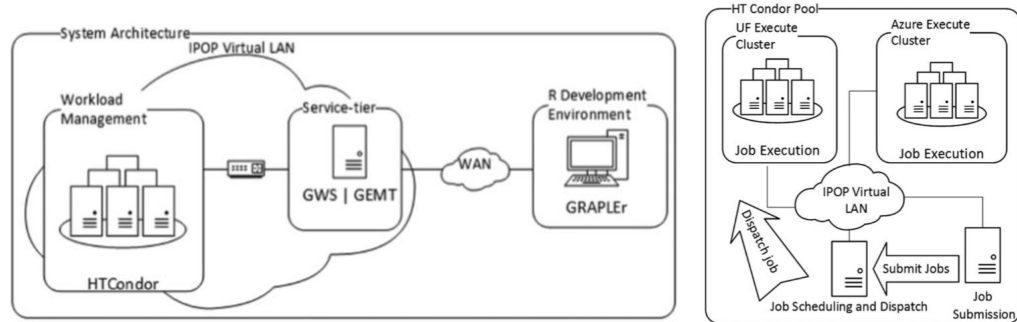
Or specification of the cluster number only as with this command be careful this will cause all jobs with in that cluster to be removed.

```
$ condor_rm 6
```

Grappler integrations

According to job that we need to code and generate jobs definition before submit to HTCondor, it quite hard to use by common users.

So, GRAPLER provide system that split R code and generate R code automatically before submit to HTCondor.



GRAPLER system architecture

Sending data back from execution machine

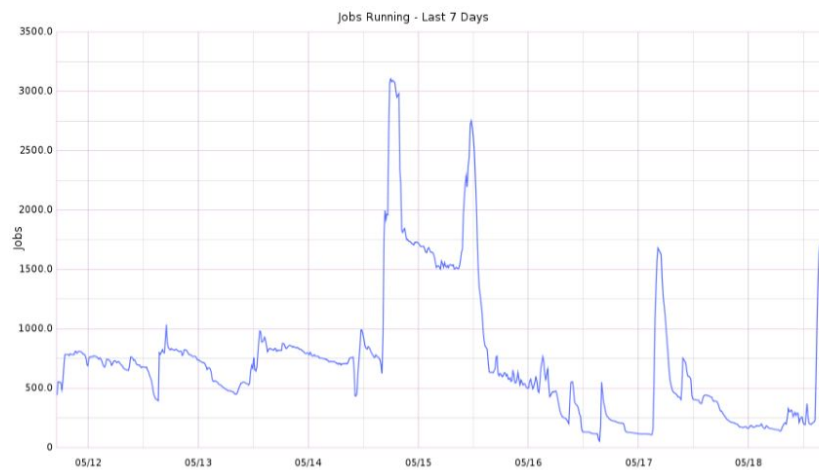
I found one tricky way to send data back from machine which we don't know where will be execute from journey paper. Here is an example.

```
#!/bin/bash

metric="htcondor.running"
value=$(condor_q | grep R | wc -l)
timestamp=$(date +%s)

echo "$metric $value $timestamp" | nc \
graphite.yourdomain.edu 2003
```

Example of echo value and pipe to netcat to graphite



Graphite shows statistic of value (from condor_q) by each timestamp

2. Grafana

Authorization management (Permissions)

Grafana supports 3 types of permission which are **Admin**, **Editor**, **Viewer** via organization, team and directly to user.

Example permissions of each role

Admin role

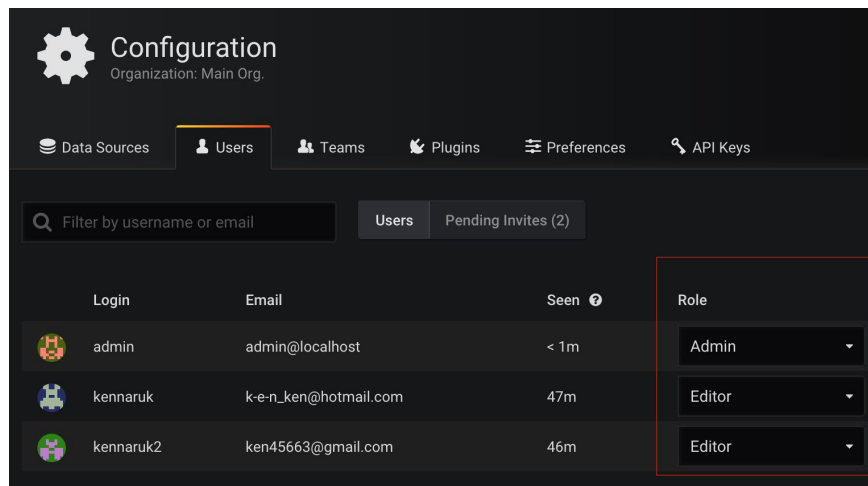
- Add & Edit data sources.
- Add & Edit organization user & teams.
- Configure App plugins & set org settings.

Editor role

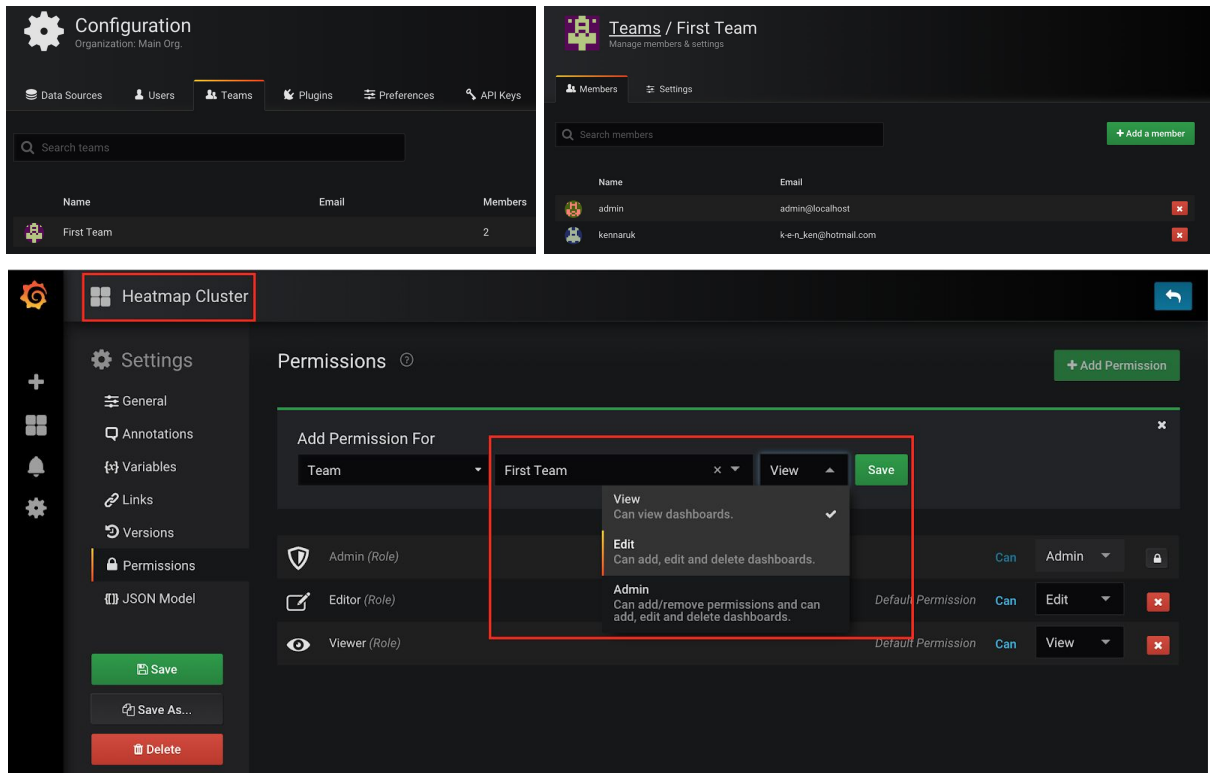
- Can create and modify dashboards & alert rule. This can be disabled on specific folders and dashboards
- Cannot create or edit data sources nor invite new users

Viewer Role.

- View any dashboard This can be disabled on specific folders and dashboards.
- Cannot create or edit dashboards not data sources



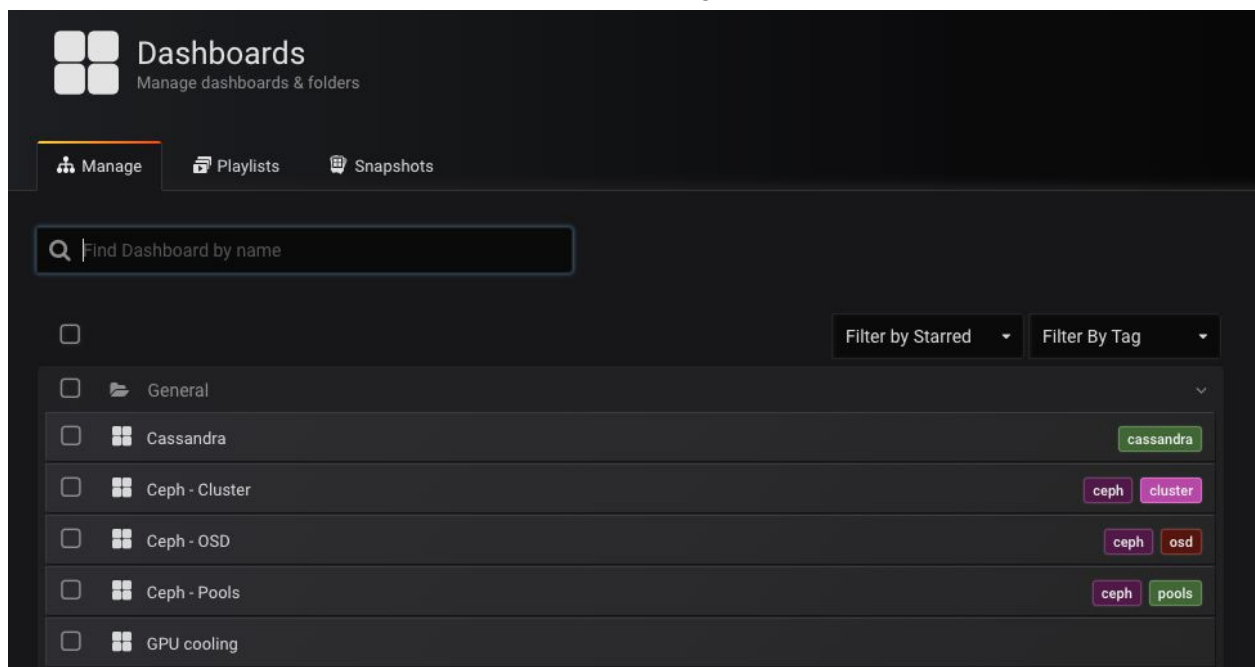
Example configure role directly to user



Example to specify configure via team or role in each dashboards

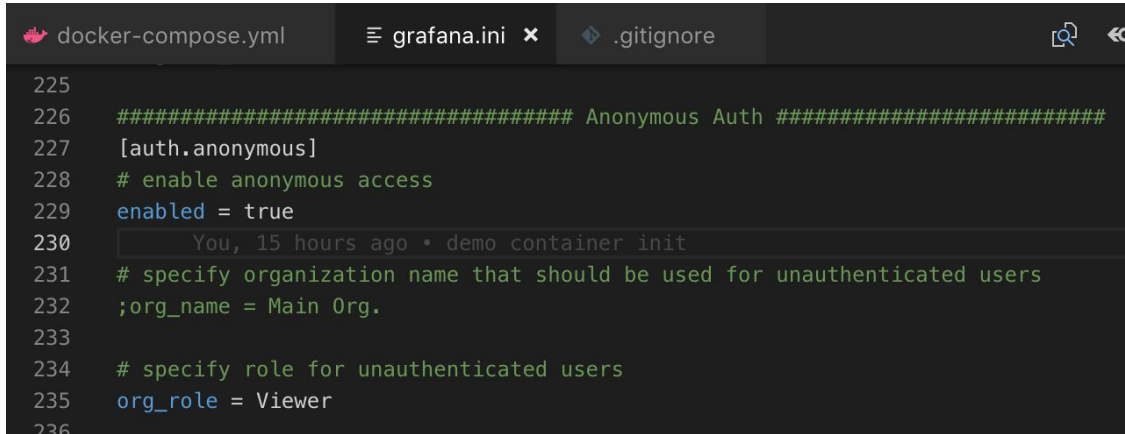
Anonymous login

According to last meeting, <https://grafana.nautilus.optiputer.net/> grafana dashboard that we can access and see default dashboards without login.



This configuration called “Anonymous login”, all we need to do is configure grafana in `grafana.ini` with 3 flags;

- **enabled**: enable anonymous login or not
- **org_name**: organization name for anonymous user
- **org_role**: or set permission to them directly



```
225
226 ##### Anonymous Auth #####
227 [auth.anonymous]
228 # enable anonymous access
229 enabled = true
230 You, 15 hours ago • demo container init
231 # specify organization name that should be used for unauthenticated users
232 ;org_name = Main Org.
233
234 # specify role for unauthenticated users
235 org_role = Viewer
236
```

I'm already deploys grafana dashboard with this mode. You can visit at <http://flynn.sci.tu.ac.th:3000/> with no https and www.

Providing default templates for non-developer users

On this topic, I still don't have any idea to do maybe I should dig deeper on grafana source code but it could consumes a lot of time to do.

Easiest way is maintain by using organization but *Each organization contains their own dashboards, data sources and configuration, and **cannot be shared between orgs**. While users may belong to more than one, multiple organization are most frequently used in multi-tenant deployments*

Variables and JSON Model

We can manage dashboard more dynamic by using variables it similar to declare normal variable in programming languages.

General			
Name	datasource	Type	Datasource
Label	optional display name	Hide	

Data source options

Type: InfluxDB

Instance name filter: /*-(*)-*/

Preview of values

NOAA_water_database default

Add

Declare new variable

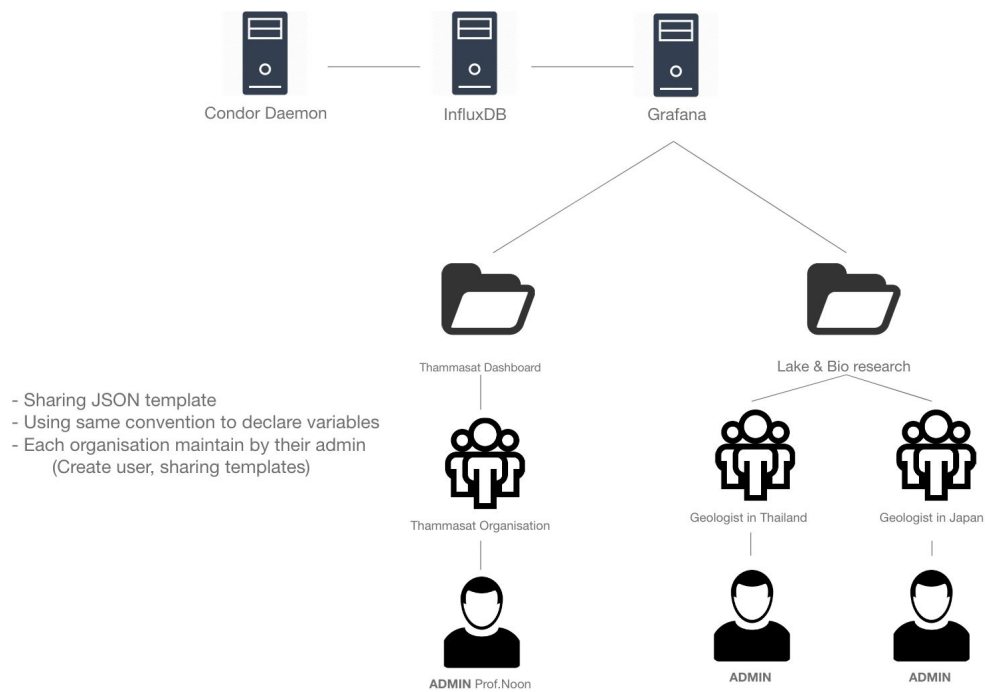
Then, we share template of each dashboard by JSON Model or JSON template which should have same convention to map each variable.

Example, we create template, share it and write document that this template requires variable named “datasource” and set to your own Datasource.

Save Changes

JSON Model or JSON template

But it still seem to be hard to use? I think. So, I don't have any idea yet.



Implement by features that grafana provides

3. References

HTCondor References

1. <https://research.cs.wisc.edu/htcondor/HTCondorWeek2015/presentations/BryantThapa-monitoring.pdf>
2. https://research.cs.wisc.edu/htcondor/tutorials/videos/2014/Intro_To_Using_HTCondor.html
3. <https://research.cs.wisc.edu/htcondor/>
4. https://www.hep.phy.cam.ac.uk/condor/condor-V8_6_3-Manual.pdf
5. <https://research.cs.wisc.edu/htcondor/manual/quickstart.html>

Grafana References

1. <http://docs.grafana.org/administration/permissions/>
2. <http://flynn.sci.tu.ac.th:3000>