KIAF 유저 가이드

버젼v0.3

GSDC

August 9, 2016

Contents

List of Figures

1 KIAF 서버 접속

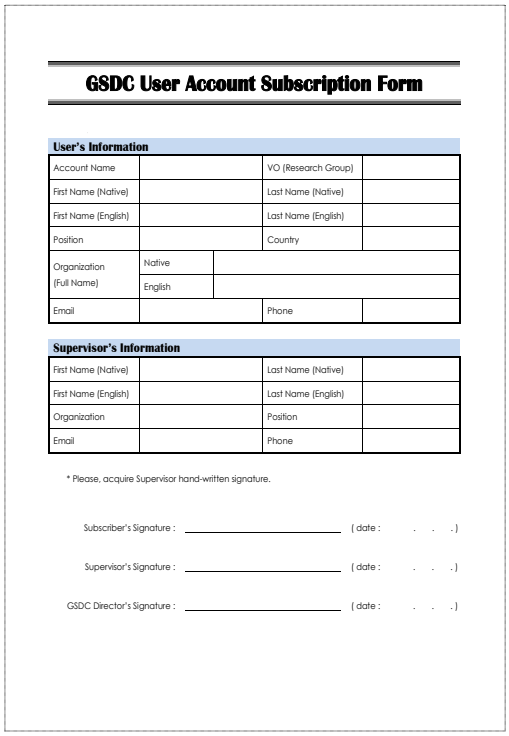
이 절에서는 KIAF의 계정을 발급 받아 KIAF에 접속하고 분석에 필요한 ALICE 환경을 로드하는 과정까지 알아보겠습니다.

1.1 사용자 계정 생성

GSDC에서 제공하는 KIAF를 이용하기 위해서는 KIAF 계정을 발급 받아야 합니다. 여기에서는 KIAF 계정을 발급받는 과정을 안내합니다.

1.1.1 계정 신청서

계정 신청서는 아래의 그림이나 부록의 계정 신청서 양식을 사용하여 작성합니다.
(계정 신청서는 계정 발급 이후 신청서는 즉시 폐기 됩니다.)



GSDC User Account Subscription Form

User's Information

Account Name		VO (Research Group)	
First Name (Native)		Last Name (Native)	
First Name (English)		Last Name (English)	
Position		Country	
Organization (Full Name)	Native		
	English		
Email		Phone	

Supervisor's Information

First Name (Native)		Last Name (Native)	
First Name (English)		Last Name (English)	
Organization		Position	
Email		Phone	

* Please, acquire Supervisor hand-written signature.

Subscriber's Signature : _____ (date : . . .)

Supervisor's Signature : _____ (date : . . .)

GSDC Director's Signature : _____ (date : . . .)

Figure 1: GSDC 사용자 계정 신청서 양식

Account Name은 사용자 ID가 될부분으로 알아보기 쉽게 작성[1]하고, 계정이 생성된 후 임시 비밀번호가 사용자 Email로 전송 됩니다.

작성한 신청서는 KISTI KIAF 담당자에게 제출합니다.

계정 요청 담당자 : `kiaf-support@kisti.re.kr`

[1]ID를 정확하게 만들기 위해 이후 신청서를 담당자에게 제출 할때 메일에 같이 기입하여 보내기를 권장

1.2 UI 서버 접속

위의 절차를 따라 계정을 요청 하여 임시 비밀번호를 받으면, 요청한 계정과 임시 비밀번호로 kiaf.sdfarm.kr에 접속이 가능 합니다. 이때 ssh로 접속이 가능한 포트는 4280번 입니다.

```
ssh -p 4280 kong91@kiaf.sdfarm.kr
```

KIAF에 처음 접속할 시 임시 비밀번호를 이용하여 로그인 하면 현재 비밀 번호를 물어본 뒤에 새로운 비밀번호를 설정 하도록 합니다. 비밀번호 규정에 맞도록 새로운 비밀번호를 정해 줍니다. 이때 비밀번호의 설정이 완료 되면 연결이 종료됩니다. 다시 접속을 시도 하여 새로운 비밀번호를 입력하면 접속에 성공합니다. 접속이 되면 사용자의 홈으로 이동 되는데, 사용자의 홈 디렉토리는 /alice/home/*계정명* 입니다.

1.3 ALICE 환경 로딩

`kiafenv` 명령어를 사용하여 KIAF에 필요한 ALICE 환경을 로드합니다.

```
[kong91@kiaf-test-01 ~]$ kiafenv
Load the ALICE environment
```

이때 로드되는 환경의 AliRoot 혹은 AliPhysics의 버젼은 $HOME/.alice_env/alice_env.conf에서 정의됩니다.

```
[kong91@kiaf-test-01 ~]$ cat $HOME/.alice_env/alice_env.conf
export ALIPHYSICS_VERSION="vAN-20160417-1"
```

위의 예시는 vAN-20160417-1 버전의 AliPhysics 설정을 위한 값 입니다. AliROOT를 사용하고자 한다면, 아래와 같이 값을 입력하면 됩니다.

```
export ALICE_ROOT_VERSION="v5-07-20-4"
```

KIAF에서 사용가능한 AliRoot 및 AliPhysics의 버젼은 `http://alimonitor.cern.ch/packages`에서 확인이 가능합니다. 사용하고자 하는 버젼의 CVMFS status가 Available인지 확인 하시기 바랍니다. 사용이 불가능한 버젼을 실행할 경우 에러가 발생하며 AliRoot가 실행 되지 않습니다.

kiafenv 명령어의 옵션으로 `-t` 또는 `--token`과 `-p` 또는 `--pod`가 존재하며, t옵션은 Alien token을 발급하고 p옵션은 PoD환경을 로드합니다. `-h` 또는 `--help`를 입력시 도움말을 출력합니다. 모든 옵션은 동시에 사용이 가능합니다. 옵션의 token과 PoD에 관해서는 각각 절6.1과 절6.2에서 설명합니다.

```
[kong91@kiaf-test-01 ~]$ kiafenv -h
This command load the ALICE environment for KIAF
options:
 -p, --pod    : Load the PoD environment
```

```
-t, --token  : Create alien session token
-h, --help   : Display help
```

2 데이터 셋

이 절에서는 데이터셋을 요청하고 확인 하는 방법에 대해 알아보겠습니다.

2.1 데이터 셋 요청

KIAF에서는 분석에 필요한 데이터를 로컬로 옮겨 놓은뒤 분석을 수행합니다. 이때 필요한 데이터 셋을 담당자에게 요청 하도록 합니다. 요청은 메일을 통해 담당자에게 필요한 데이터셋의 정보를 보내주시면 됩니다. 데이터셋의 정보는 Data:Sim, Period, Run Number, Variant, Pass 등 입니다. 요청하신 데이터 셋은 /xrootfs/alice 아래에 Alien Catalogue와 같은 경로로 저장 되어 있습니다.

예 - /xrootdfs/alice/alice/sim/2013/LHC13d3/195675/AOD159/0001/root_archive.zip

데이터셋 요청 담당자 : `kiaf-data-request@kisti.re.kr`

위 메일은 데이터셋을 요청할 때에만 이용하고, 일반적인 문의는 `kiaf-support@kisti.re.kr`를 이용하기를 요청 드립니다.

2.2 데이터 셋 확인

위에서 언급한 것과 같이 KIAF에 저장되어 있는 데이터는 /xrootdfs/alice 아래에 위치합니다. 직접 디렉토리에 접근하여 파일을 확인 할수 있습니다. 또한 요청한 데이터 셋의 파일 리스트는 /pool/datalist 안에 type_period_run_pass_variant.txt의 형식으로 존재합니다.

예 - sim_LHC13d3_195675_AOD159.txt

또한 명령어 `showdataset`을 통해 스테이징 현황을 알 수 있습니다. 명령어와 함께 검색을 원하는 데이터 셋의 이름을 적어주면 해당하는 데이터셋만 보여줍니다. 데이터셋을 적어주지 않으면 전체를 조회하며, 조회되는 양에 따라 많은 시간이 소요 될수 있습니다.

```
[kong91@kiaf-test-01 ~]$ showdatalist LHC12i
Dataset                                          Queued  Downloading  Success  Fail
data/2012/LHC12i/000192762/ESDs/muon_calo_pass2  0       0            220      0
data/2012/LHC12i/000192765/ESDs/muon_calo_pass2  0       0            480      0
data/2012/LHC12i/000192772/ESDs/muon_calo_pass2  0       0            1920     0
```

3 HTCondor 사용법

이절에서는 분석에 사용 되는 HTCondor의 기본적인 사용법에 대해 소개하고자 합니다.

3.1 Condor cluster 상태 확인

condor_status는 현재 condor 슬롯의 상황을 보여주는 명령어입니다.

```
[kong91@kiaf-test-01 ~]$ condor_status
Name               OpSys      Arch   State     Activity LoadAv Mem     ActvtyTime

slot1@kiaf-test-02 LINUX      X86_64 Unclaimed Idle      0.110 129013 18+11:37:38
slot1@kiaf-test-03 LINUX      X86_64 Unclaimed Idle      0.080 129013 18+11:37:29
slot1@kiaf-test-04 LINUX      X86_64 Unclaimed Idle      0.070 129013 18+11:37:24
slot1@kiaf-test-05 LINUX      X86_64 Unclaimed Idle      0.090 129013 18+11:37:27
                     Machines Owner Claimed Unclaimed Matched Preempting

        X86_64/LINUX        4     0       0         4       0          0

               Total        4     0       0         4       0          0
```

3.2 Condor job Queue 확인

condor_q 명령어는 요청 한 잡의 queue를 보여 줍니다. 잡이 수행 중에 I나 H의 상대로 변하면 로그를 확인 하여 잘못 된 부분을 수정해야 합니다. 잡의 상태 뿐만 아니라 ID, OWNER 등도 확인이 가능 합니다.

```
[kong91@kiaf-test-01 multi_output]$ condor_q


-- Schedd: kiaf-test-01.sdfarm.kr : <134.75.125.139:14337?...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
9618.0   kong91          6/13 16:58   0+00:00:08 R  0   0.3  condor_dagman -p 0
9619.0   kong91          6/13 16:59   0+00:00:00 I  0   0.0  job_wrap.sh AliPhy
9620.0   kong91          6/13 16:59   0+00:00:00 I  0   0.0  job_wrap.sh AliPhy
9621.0   kong91          6/13 16:59   0+00:00:00 I  0   0.0  job_wrap.sh AliPhy
9622.0   kong91          6/13 16:59   0+00:00:00 I  0   0.0  job_wrap.sh AliPhy
9623.0   kong91          6/13 16:59   0+00:00:00 I  0   0.0  job_wrap.sh AliPhy


6 jobs; 0 completed, 0 removed, 5 idle, 1 running, 0 held, 0 suspended
```

3.3 잘못된 잡 삭제

작업에 문제가 있어 잡이 중지되었을때 강제로 잡을 삭제하는 명령어입니다. condor_rm *ID*를 입력 하여 수행 합니다.

```
[kong91@kiaf-test-01 final_test]$ condor_rm 9629
All jobs in cluster 9629 have been marked for removal
```

4 시뮬레이션 분석(준비중)

준비중입니다.

5 사용자 예제

새롭게 구축된 KIAF는 HTCondor라는 배치 시스템을 이용합니다. 이 절에서는 예제 코드 사용법과 예제 코드를 이용한 사용자 분석 방법에 대해 알아 보겠습니다.

5.1 예제 파일

5.1.1 예제1. Batch_test

예제 파일은 /tutorial 아래에 있습니다. batch_test 폴더를 복사하거나, batch_test.tar파일을 복사 한뒤 압축을 풀어 사용하면 됩니다.

```
cp -r /tutorial/batch_test $HOME/batch_test
```

또는

```
cp -r /tutorial/batch_test.tar $HOME/batch_test.tar
tar -xvf batch_test.tar
```

batch_test 폴더 안의 파일 리스트는 아래와 같습니다.

```
[kong91@kiaf-test-01 batch_test]$ ls -l
total 28
-rw-r--r--. 1 kong91 kong91  776 Jun  7 10:06 AddMyTask.C
-rw-r--r--. 1 kong91 kong91 2036 Jun  7 10:06 AliAnalysisTaskMyTask.cxx
-rw-r--r--. 1 kong91 kong91  895 Jun  7 10:06 AliAnalysisTaskMyTask.h
-rw-r--r--. 1 kong91 kong91 1120 Jun  7 10:06 runAnalysis.C
-rwxrw-r--. 1 kong91 kong91 3546 Jun  7 12:56 runCondor.sh
-rw-rw-r--. 1 kong91 kong91 5325 Jun  7 10:06 sim_LHC13d3_195675_AOD159.txt
```

위 리스트에서 확인 하였듯이 batch_test폴더 안에는 분석에 사용되는 여러 코드와 배치 잡을 실행 시켜주기 위한 매크로 파일이 있습니다. 실제 잡을 실행 시키기 위해서는 runCondor.sh 파일을 실행 시키면 됩니다. 예시 파일은 수정 없이 바로 실행 해도 에러없이 실행이 가능 합니다.

이때 runCondor.sh는 runAnalysis.C의 인풋 파일을 작게 나누어 병렬 계산을 도와주는 매크로 입니다. 즉, runAnalysis.C 매크로 자체의 에러가 발생한다면 runCondor.sh에서도 에러가 발생하니 local test를 먼저 수행하길 권장합니다.

`./runCondor.sh` 명령을 통해 스크립트를 실행하면 아래와 같은 결과를 터미널을 통해 볼 수 있습니다.

```
[kong91@kiaf-test-01 batch_test]$  ./runCondor.sh
Renaming rescue DAGs newer than number 0
------------------------------------------------------------------------
```

```
File for submitting this DAG to Condor  : 160612173845/macro/condor.dag.condor.sub
Log of DAGMan debugging messages        : 160612173845/macro/condor.dag.dagman.out
Log of Condor library output            : 160612173845/macro/condor.dag.lib.out
Log of Condor library error messages    : 160612173845/macro/condor.dag.lib.err
Log of the life of condor_dagman itself : 160612173845/macro/condor.dag.dagman.log

Submitting job(s).
1 job(s) submitted to cluster 9244.
-----------------------------------------------------------------------
```

이때 160612173845는 잡을 수행시킨 년월일시분초의 이름로 만들어진 폴더 이며, 워킹 디렉토리로 사용 됩니다. 각 잡의 로그는 워킹디렉토리 아래 logs폴더에 위치합니다. 최종 결과 파일은 batch_test폴더 안에 merge_AnalysisResults.root라는 이름으로 생성 됩니다.

수행중인 잡의 큐는 `condor_q` 라는 명령어를 통해 확인이 가능 합니다.

```
[kong91@kiaf-test-01 batch_test]$ condor_q


-- Schedd: kiaf-test-01.sdfarm.kr : <134.75.125.139:14337?...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
9317.0   kong91          6/13 00:15   0+00:00:08 R  0   0.3  condor_dagman -p 0
9318.0   kong91          6/13 00:15   0+00:00:01 R  0   0.0  job_wrapper.sh Ali
9319.0   kong91          6/13 00:15   0+00:00:01 R  0   0.0  job_wrapper.sh Ali
9320.0   kong91          6/13 00:15   0+00:00:01 R  0   0.0  job_wrapper.sh Ali
9321.0   kong91          6/13 00:15   0+00:00:01 R  0   0.0  job_wrapper.sh Ali
9322.0   kong91          6/13 00:15   0+00:00:01 R  0   0.0  job_wrapper.sh Ali
```

5.1.2 예제2. Multi_output

이번에는 분석 결과로 여러개의 아웃풋 파일이 생성 되는 예제 입니다.

예제 파일은 /tutorial 아래에 있습니다. multi_output 폴더를 복사하거나, multi_output.tar파일을 복사 한뒤 압축을 풀어 사용하면 됩니다. 위의 예제와 같이 구성 되어 있습니다. `./runCondor.sh` 명령을 통해 스크립트를 실행하면 2개의 결과 파일이 생성 되는 것이 확인 가능 합니다.

```
[kong91@kiaf-test-01 multi_output]$ ls -ls *root
26 -rw-r--r--. 1 kong91 kong91 4265 Jun 13 16:34 merge0_AnalysisResults.root
26 -rw-r--r--. 1 kong91 kong91 4265 Jun 13 16:34 merge1_AnalysisResults.root
```

5.2 사용자 분석 코드 입력 방법

이번에는 위에서 수행 한 예제 파일을 기반으로 사용자 분석코드를 수행하는 방법에 대해 알아보도록 하겠습니다. 진행하기에 전에 앞의 예제 부분을 숙지하길 권장합니다.

5.2.1 분석1. Batch_test

우선 사용자가 로컬모드로 테스트가 완료된 매크로를 준비합니다. 이때 사용 되는 C, cxx ,h 파일등 모든 파일이 필요합니다. 그리고 예제 파일 중에 runCondor.sh파일을 복사해 옵니다.

runCondor.sh 파일의 윗부분은 아래와 같습니다.

```
#!/bin/bash
condor_slot=40
nfile=10

work_dir=‘date +%y%m%d%H%M%S‘
run_macro="runAnalysis.C"
data_file="sim_LHC13d3_195675_AOD159.txt"
input_files="AddMyTask.C,AliAnalysisTaskMyTask.cxx,AliAnalysisTaskMyTask.h"
out_file="AnalysisResults.root"
```

- **condor_slot** condor에 동시에 할당될 슬롯의 최대값 입니다. 현재 KIAF에는 160개의 슬롯이 준비되어 있으므로 다른 사용자들의 원할한 사용을 위하여 40의 값을 권장 합니다.
- **nfile** 슬롯당 분배 되는 인풋 파일의 갯수 입니다. Grid잡에서 CreateAlienHandler.C의 SetSplitMaxInputFileNumber의 값과 동일한 역할을 합니다. `cat data_file | wc -l` 명령어[2] 를 통해 전체 인풋 파일의 갯수를 알수 있으니, 이를 토대로 적당한 값을 사용자가 기입 합니다.
- **work_dir** 생성 되는 워킹 디렉토리의 이름으로 사용자가 원하는 이름으로 변경하여도 무방합니다.
- **run_macro** aliroot를 통해 실행 되는 분석 매크로 입니다.
- **data_file** 실제 입력 되는 데이터가 아닌 데이터의 경로 리스트 파일 입니다. 요청하셨던 데이터 파일의 리스트는 /pool/datalist아래에 런별로 위치하며, 복사하여 사용하시거나 따로 만들어 사용 하시면 됩니다. 다만 데이터 리스트 파일은 runCondor.sh와 같은 경로에 위치하여야 합니다. 그렇지 않은 경우 여기에는 절대경로의 파일명을 적어 주셔야 합니다.
- **input_files** 실제 분석에 사용 되는 C, cxx ,h 파일등 모든파일의 명을 적어 주시면 됩니다. 마찬가지로 runCondor.sh와 같은 경로에 위치하지 않은 경우 절대경로를 적어 주셔야 합니다. 파일은 콤마(,)로 구분하여 여러개를 적어주셔도 무방 합니다.
- **out_file** 결과를 통해 얻는 root파일의 이름으로 현재는 1개의 결과 파일만 생성이 가능 합니다. 최종 결과 파일은 runCondor.sh와 같은 위치에 merge_out_file명으로 생성 됩니다.

[2] |는 엔터키 위의 ₩(또는 /)키를 shift와 같이 누르면 입력이 가능 합니다.

5.2.2 분석2. Multi_output

이번에는 분석 결과로 여러개의 아웃풋 파일이 생성 되는 경우를 위한 매크로 입니다. 위 예제와 같이 테스트가 완료된 매크로를 준비 합니다. 매크로가 준비 되었을때 수정해야할 파일은 **runCondor.sh** 파일 입니다.

runCondor.sh 파일의 윗부분은 아래와 같습니다.

```
condor_slot=40
nfile=10


work_dir=‘date +%y%m%d%H%M%S‘
run_macro="runAnalysis.C"
data_file="sim_LHC13d3_195675_AOD159.txt"
input_files="AddMyTask.C,AliAnalysisTaskMyTask.cxx,AliAnalysisTaskMyTask.h"
out_files="AnalysisResults.root, AnalysisResults.root"
```

분석1과 달라진 부분은 **out_files**입니다. 이곳에 생성 아웃풋 파일의 이름을 콤마(,)로 구분 하여 적어주면, 최종 결과 파일은 runCondor.sh와 같은 위치에 merge#_out_file명[3]으로 생성 됩니다.

[3] #은 merge 번호로 out_files 에 적어준 순서로 0부터 차례로 증가합니다.

6 사용자 분석 환경에 따른 추가사항

6.1 Alien token

6.1.1 인증서 복사

$HOME/.globus 디렉토리에 개인 인증서 파일인 usercert.pem, userkey.pem을 복사합니다.
개인 머신에서 `scp` 명령어를 사용하여 복사 하면됩니다.

```
kong91:~/> scp -r -P 4280 $HOME/.globus kong91@kiaf.sdfarm.kr:/alice/home/kong91/.
kong91@kiaf.sdfarm.kr's password:
usercert.pem                  100% 3334     3.3KB/s   00:00
userkey.pem                   100% 1907     1.9KB/s   00:00
```

6.1.2 Alien Token 발급

토큰은 `alien-token-init` 명령어를 통해 발급이 가능합니다. 다만 KIAF 계정명과 CERN 계정명이 일치하지 않는 경우, `alien-token-init` 뒤에 CERN계정명을 적어 줘야 합니다.

또한 `kiafenv -t` 를 통한 토큰 발급을 위해 $HOME/.alice_env/alice_env.conf에 한줄을 추가하여, export alien_API_USER=*CERN 계정명*을 적어주시기 바랍니다.

6.1.3 Condor 워커노드에 Alien Token 발급

분석 코드중에 토큰이 필요한 경우에는 워커노드에도 alien token이 필요합니다. 이때는 `alien-token-init` 명령이 아닌 `kiafenv -t` 명령을 이용 하여 토큰을 발급 받습니다. 처음 사용시에는 추가적인 설정이 필요합니다. 관리자에게 연락하여 추가 설정을 요청하길 바랍니다. 사용자의 원할한 분석작업을 위하여 계정을 발급 받고 관리자에게 연락하여 절차를 수행하시길 권장 합니다.

6.2 PROOF 사용법

6.2.1 PROOF-Lite 사용법

PROOF-Lite는 특별한 설정 없이 AliROOT의 명령을 이용해 실행이 가능 합니다.(PROOF-Lite는 팜전체가 아닌 UI머신의 모든 CPU를 사용합니다.) AliROOT를 실행한 후 Proof 세션을 오픈합니다.

```
[kong91@kiaf-test-01 ~]$ aliroot -l
root [0] TProof::Open("")
 +++ Starting PROOF-Lite with 40 workers +++
Opening connections to workers: OK (40 workers)
Setting up worker servers: OK (40 workers)
PROOF set to parallel mode (40 workers)
(class TProof*)0x2d20420
root [1]
```

UI의 CPU 40개를 모두 사용하여 40개의 워커가 준비되었음을 확인 할 수 있습니다.

6.2.2 PoD 사용법

PoD는 Proof를 사용자가 원하는 대로 설정하여 사용하도록 도와주는 어플리케이션 입니다. PoD를 사용하면 팜 전체의 CPU자원을 활용 할 수 있습니다.

PoD 환경 설정

PoD 환경 설정은 `kiafenv -p` 명령을 이용합니다. 마지막 두줄은 PoD를 처음 실행 할때에만 나타나며 이후에는 보이지 않습니다. (kiafenv를 두번 실행 할 경우 aliroot를 시작할때 warning 문구가 나타 날 수 있습니다.)

```
[kong91@kiaf-test-01 ~]$ kiafenv -p
Load the ALICE environment
Set the AliPhysics::vAN-20160417-1 environment
Load PoD environment
Generating a default PoD configuration file...
Generating a default PoD configuration file - DONE.
```

위 명령을 수행하면 사용자 디렉토리 아래에 .PoD의 폴더가 생성 됩니다. 폴더 내의 PoD.cfg파일을 수정하여 세부 조정이 가능 합니다. 파일을 수정한 뒤에는 pod환경을 다시 불러와야 합니다. 이때는 `source /pool/PoD/3.16/PoD_env.sh`명령을 사용합니다.

PoD 실행 방법

설정을 마친 뒤 PoD를 사용하기 위해서는 몇가지 절차가 필요합니다. 우선 PoD 서버를 시작해야 합니다. `pod-server start`명령어를 이용합니다.

```
[kong91@kiaf-test-01 ~]$ pod-server start
Starting PoD server...
updating xproofd configuration file...
starting xproofd...
starting PoD agent...
preparing PoD worker package...
select user defined environment script to be added to worker package...
selecting pre-compiled bins to be added to worker package...
PoD worker package: /share/kong91/.PoD/wrk/PoDWorker.sh
-------------------------
XPROOFD [2860349] port: 21002
PoD agent [2860374] port: 22002
PROOF connection string: kong91@kiaf-test-01.sdfarm.kr:21002
-------------------------
```

위 처럼 PoD 서버가 정상적으로 시작이 되면 원하는 만큼 PROOF 워커를 불러 옵니다. 여기에 사용 되는 명령어는 `pod-submit -r condor -n 10` 입니다.

```
[kong91@kiaf-test-01 ~]$ pod-submit -r condor -n 10
Job ID: 17606
```

`-r` 옵션은 PROOF가 사용할 배치시스템을 지정하는 것입니다. KIAF에서는 condor 만 지원 하기 때문에 이 값은 변경할 수 없습니다. `-n` 옵션은 실행할 PROOF 워커의 갯수입니다. condor를 사용하기 때문에 최대 갯수는 160개 입니다. 또한 PROOF를 condor를 통해 실행 하기 때문에 condor의 잡 ID를 할당 받습니다.

PoD가 준비가 되었는지는 `pod-info -n`으로 확인합니다. 준비가 완료된 PROOF의 갯수를 표기합니다. condor를 실행하는데 약간의 시간이 필요함으로 지정한 갯수와 맞지 않는 경우 잠시 기다립니다.

```
[kong91@kiaf-test-01 ~]$ pod-info -n
10
```

condor 명령어인 `condor_q`를 통해서도 확인 할 수 있습니다. PoD 워커가 실행이 가능하면 condor의 상태가 R로 표시 됩니다.

```
[kong91@kiaf-test-01 ~]$ condor_q
```

```
-- Schedd: kiaf-test-01.sdfarm.kr : <134.75.125.139:12198?...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
17606.0   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.1   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.2   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.3   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.4   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.5   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.6   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.7   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.8   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
17606.9   kong91          6/23 13:37   0+00:10:12 R  0   2.0  PoDWorker.sh
```

이렇게 모든 준비를 마치면 AliROOT에서 PoD에 접속하여 사용 하면 됩니다.

```
[kong91@kiaf-test-01 ~]$ aliroot -l
root [0] TProof::Open("pod://","masteronly")
Starting master: opening connection ...
Starting master: OK
Note: File "iostream" already loaded
PROOF set to sequential mode
(class TProof*)0x31e6f80
root [1]
```

참고 자료

[1] HTCondor 입문 (2016 GSDC School of Data Grid Computing). https://indico.cern.ch/event/476290/contributions/1154545/attachments/1221990/1786975/HTCondor___.pdf.

[2] PROOF tutorial. https://root.cern.ch/root/htmldoc/tutorials/proof/index.html.

[3] PoD User's Manual. http://pod.gsi.de/doc/3.16/PoD.pdf.

A GSDC 사용자 계정 신청서 양식

다음페이지의 양식을 출력하여 작성 합니다. 작성해야할 내용은 아래와 같습니다.

1. **User's Information** 계정 발급을 요청하는 사용자의 정보를 입력합니다.

   - **Account Name** 사용자에게 발급 될 ID 입니다.
     다른 사용자가 사용중인 경우를 대비해 2개 이상 작성하기를 권장합니다.
     (작성 순서에 따라 발급을 시도 합니다.)
   - **VO (Research Group)** 현재 소속된 실험그룹을 입력합니다.
     KIAF를 사용하는 실험 그룹은 ALICE 입니다.
   - **Position** 현재 사용자의 직위를 입력합니다.
     (석사과정 학생, 박사과정 학생, 박사후 연구원 등)
   - **Country** 현재 소속된 기관이 위치한 국가를 입력합니다.
   - **Organization** 현재 소속된 기관을 기입합니다.
   - **Organization Unit** 현재 소속된 기관의 부서를 기입합니다. (물리학과 등)
   - **Email** 연락이 가능한 이메일 주소
   - **Phone** 연락이 가능한 연락처

2. **Supervisor's Information** 위에 기입한 사용자를 지도하는 분의 정보를 입력합니다.

   - **Organization** 현재 소속된 기관을 기입합니다.
   - **Position** 지도교수님의 직위를 입력합니다. (교수 등)
   - **Email** 연락이 가능한 이메일 주소
   - **Phone** 연락이 가능한 연락처

사용자 본인의 서명을 하고 지도교수님의 서명을 받아 스캔하여 이메일로 제출 합니다.(`kiaf-support@kisti.re.kr`)

GSDC User Account Subscription Form

User's Information

Account Name		VO (Research Group)	
First Name (Native)		Last Name (Native)	
First Name (English)		Last Name (English)	
Position		Country	
Organization (Full Name)	Native		
	English		
Email		Phone	

Supervisor's Information

First Name (Native)		Last Name (Native)	
First Name (English)		Last Name (English)	
Organization		Position	
Email		Phone	

* Please, acquire Supervisor hand-written signature.

Subscriber's Signature : _____ (date : . . .)

Supervisor's Signature : _____ (date : . . .)

GSDC Director's Signature : _____ (date : . . .)

B KIAF 구성

1. **컴퓨팅 자원**
   - **서버** 총 5대
   - **CPU** 서버당 20코어 총 100 물리 코어(하이퍼 스레딩을 적용하여 200코어)
   - **RAM** 서버당 128GB
   - **저장장치** NAS 200TB
   - **네트워크** 10Gbps
2. **소프트웨어**
   - **운영 체제** Scientific Linux 6
   - **HTCondor** v8.4.7
   - **PoD** v3.16
   - **XRootD** v4.3.0
3. **Condor 구성** UI / CE - 1대, WN - 4대
4. **사용자 인증** LDAP 기반

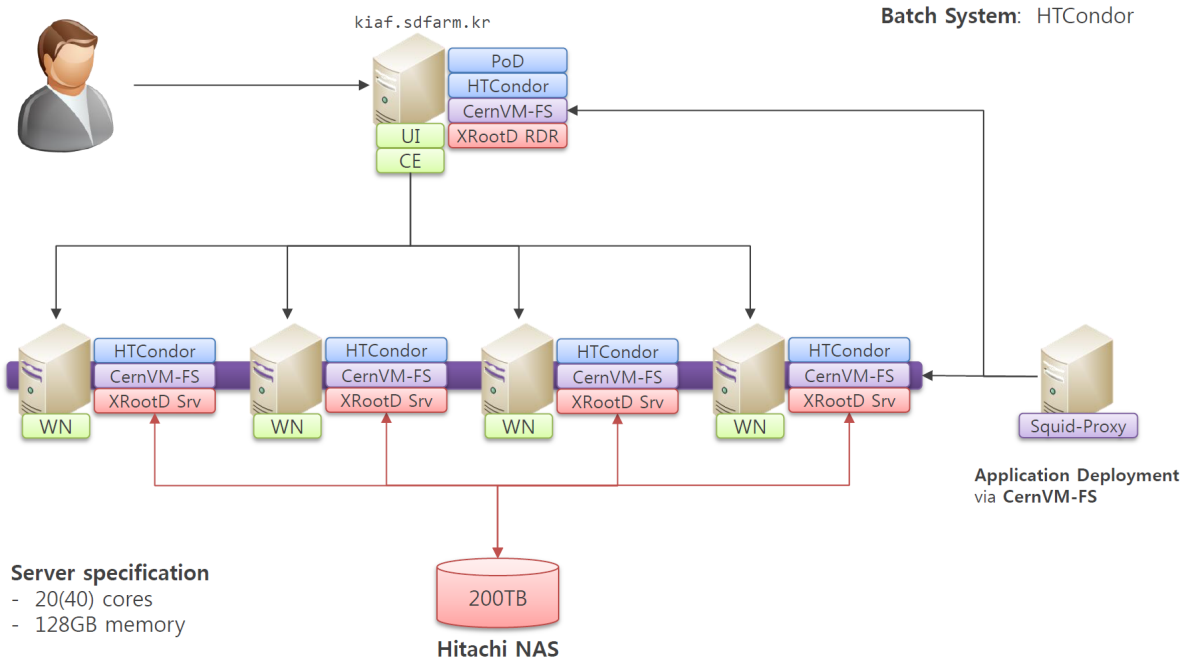


Figure 2: KIAF 구성도