Table of Contents

CRAB tutorial (advanced)

Complete:	Go to SWGuideCrab

Exercises

The tips you can see with the `show` links are in progressive order of difficulty. If you get stuck on one exercise try to solve it showing only the first tip before looking at the second one.

Exercise 1 - dry run

The following work area was prepared for this exercise:
`/afs/cern.ch/cms/ccs/wm/scripts/Crab/CRAB3/AdvTutorial/Exercise_1`. It contains

- the CMSSW parameter set configuration files `PSet.py` and `PSet.pkl`;
- the CMSSW release area `CMSSW_7_2_3_patch1` with additional packages used by the above CMSSW parameter set.

In order to use this already prepared work area with its full enironment, execute `cmsenv` directly inside `/afs/cern.ch/cms/ccs/wm/scripts/Crab/CRAB3/AdvTutorial/Exercise_1/CMSSW_7_2_3_patch1/src/`. Then you can setup CRAB3 and execute CRAB command from any other location you wish (for example your home area); you will only need to copy the `PSet` files to that location.

I am trying to analyze
`/VBF_HToTauTau_M-125_13TeV-powheg-pythia6/Phys14DR-PU20bx25_tsg_PHYS14_25_V1-v2/MINIAODSIM`
with the following splitting parameters:

```
config.Data.splitting = 'FileBased
config.Data.unitsPerJob = 1
```

and after waiting one day I am getting the following output with CRAB status:

```
Error Summary:

1 jobs failed with exit code 50664:

    1 jobs failed with following error message: (for example, job 1)

        Not retrying job due to wall clock limit (job automatically killed on the worker node)
```

1.A) Can you tell me what happened to my jobs?

Show Hide answer.
```
By default jobs cannot run more than 24 hours on the Grid worker nodes. They are automatically ki
time) exceeds this limit. That's what happened to the jobs; they were killed, because they reache
```

1.B) Prepare a CRAB configuration to analyze this dataset and execute `crab submit --dryrun`.

Help:

Show Hide CRAB configuration file.
```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_May2015_Exercise1'
```

```
config.JobType.psetName = 'PSet.py'
config.JobType.pluginName = 'Analysis'

config.Data.inputDataset = '/VBF_HToTauTau_M-125_13TeV-powheg-pythia6/Phys14DR-PU20bx25_tsg_PHYS1
config.Data.splitting = 'FileBased'
config.Data.unitsPerJob = 1

config.Site.storageSite = <site where you have permission to write>
```

1.C) How much time your code is taking for each event (use CRAB3 estimation of `--dryrun`)?

Show Hide answer.

```
The estimated time per event is 3 seconds.
```

1.D) Try to come up with better splitting parameters in the CRAB configuration: your target for this exercise are jobs running for 8 hours (estimate the number of events per lumi from DAS). Use 'crab submit --dryrun' (and please don't submit the task!).

Help:

Show Hide average number of events per lumi.

```
Looking at the first file in the dataset, it has in average ~100 events per lumi. One way to get

$ python
>>> lumis = [[3967, 3969], [3973, 3973], ...]
>>> sum([y-x+1 for x, y in lumis])
431
>>> 42902 / 431
99

[1] https://cmsweb.cern.ch/das/request?input=lumi%20file%3D/store/mc/Phys14DR/VBF_HToTauTau_M-125
```

Show Hide CRAB configuration file.

```
eventsPerLumi = 100
timePerEv = 4
desiredTime = 8*60*60
config.Data.unitsPerJob = desiredTime / (eventsPerLumi*timePerEv)
```

Exercise 2 - user input files

2.A) Run a task similar to the one in section Running CMSSW analysis with CRAB on Data of the CRAB3 (basic) tutorial, but with each job analyzing exactly one input file and with the task analyzing in total five input files. Also, don't use any lumi-mask nor run-range.

Help:

Show Hide CMSSW parameter-set configuration file.

```
import FWCore.ParameterSet.Config as cms

process = cms.Process('NoSplit')

process.source = cms.Source("PoolSource", fileNames = cms.untracked.vstring())
process.maxEvents = cms.untracked.PSet(input = cms.untracked.int32(10))
process.options = cms.untracked.PSet(wantSummary = cms.untracked.bool(True))
process.output = cms.OutputModule("PoolOutputModule",
    outputCommands = cms.untracked.vstring("drop *", "keep recoTracks_*_*_*"),
    fileName = cms.untracked.string('output.root'),
)
process.out = cms.EndPath(process.output)
```

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_June2015_Exercise2A'
config.General.transferOutputs = True
config.General.transferLogs = False

config.JobType.pluginName = 'Analysis'
config.JobType.psetName = 'pset_tutorial_analysis.py'

config.Data.inputDataset = '/SingleMu/Run2012B-13Jul2012-v1/AOD'
config.Data.splitting = 'FileBased'
config.Data.unitsPerJob = 1
config.Data.totalUnits = 5
config.Data.publication = True
config.Data.outputDatasetTag = config.General.requestName

config.Site.storageSite = <site where you have permission to write>
```

2.B) Look at what were the five input files analyzed by task A and create a local text file containing the LFNs of these five files. Submit a task to analyze the same input files as task A, but instead of specifying the input dataset use the local text file you just created. Once task A and task B have finished, check that both have published a similar output dataset. The goal of exercises 2.A and 2.B is to show that it is possible to run CRAB over published files treating them as "user input files" (i.e. using `Data.userInputFiles` instead of `Data.inputDataset`), although this is of course not recommended.

Note: When running over user input files CRAB will not try to find out at which sites the input files are stored. Instead CRAB will submit the jobs to the less busy site. If the input files are not stored at these sites, they will be accessed via Xrootd. Since Xrootd is of course less efficient than direct access, it is recommended to force CRAB to submit the jobs to the sites where the input files are stored by whitelisting those sites.

Help:

Show Hide how to know which input files were analyzed by task A?

```
Look for example in the job log files linked from the monitoring pages. In each job log file, at
were the input files analyzed by the corresponding job.

Answer:
/store/data/Run2012B/SingleMu/AOD/13Jul2012-v1/0000/008DBED0-86D3-E111-AEDF-20CF3019DF17.root
/store/data/Run2012B/SingleMu/AOD/13Jul2012-v1/0000/00F9715A-A1D3-E111-BE6F-E0CB4E29C4D1.root
/store/data/Run2012B/SingleMu/AOD/13Jul2012-v1/0000/00100164-41D4-E111-981B-20CF3027A5AF.root
/store/data/Run2012B/SingleMu/AOD/13Jul2012-v1/0000/0093BEF2-A4D3-E111-A6B9-E0CB4E19F979.root
/store/data/Run2012B/SingleMu/AOD/13Jul2012-v1/0000/0009F1CC-0DD4-E111-974D-20CF305B0584.root
```

Show Hide how to know at which sites are these input files stored?

```
Search for the input dataset /SingleMu/Run2012B-13Jul2012-v1/AOD in DAS.

Answer:
T2_BE_UCL, T2_IT_Legnaro, T2_RU_JINR, T1_US_FNAL, T1_IT_CNAF
```

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_June2015_Exercise2B'
config.General.transferOutputs = True
config.General.transferLogs = False

config.JobType.pluginName = 'Analysis'
config.JobType.psetName = 'pset_tutorial_analysis.py'

config.Data.outputPrimaryDataset = 'SingleMu'
```

```
config.Data.userInputFiles = open('CRAB3_Advanced_Tutorial_June2015_Exercise2B_inputFiles.txt').r
config.Data.splitting = 'FileBased'
config.Data.unitsPerJob = 1
config.Data.totalUnits = 5
config.Data.publication = True
config.Data.outputDatasetTag = config.General.requestName

config.Site.storageSite = <site where you have permission to write>
config.Site.whitelist = ['T2_BE_UCL', 'T2_IT_Legnaro', 'T2_RU_JINR']
```

2.C) Run a task as in 2.A, but turning off the publication.

2.D) Run a task over the output files from task C (the output files from task C are not published; so this is a typical case of user input files) using a local text file containing the LFNs of the output files from task C. Whitelist the site where you stored the output files from task C.

Help:

Show Hide how to get the LFNs of the output files from task C?
`Use 'crab getoutput --dump'.`

Exercise 3 - recovery task

Assume the following CRAB configuration, which uses the same CMSSW parameter-set configuration as in exercise 2, and produces 213 jobs:

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_June2015_Exercise3C'
config.General.transferOutputs = True
config.General.transferLogs = False

config.JobType.pluginName = 'Analysis'
config.JobType.psetName = 'pset_tutorial_analysis.py'
# Assume for this exercise that the default job runtime limit is 1 hour.
config.JobType.maxJobRuntimeMin = 60

config.Data.inputDataset = '/SingleMu/Run2012B-13Jul2012-v1/AOD'
config.Data.splitting = 'LumiBased'
config.Data.unitsPerJob = 240
config.Data.lumiMask = 'https://cms-service-dqm.web.cern.ch/cms-service-dqm/CAF/certification/Col
config.Data.publication = True
config.Data.outputDatasetTag =  config.General.requestName

config.Site.storageSite = <site where the user has permission to write>
```

3.A) Imagine the following situation. After submitting the task you did `crab status` and you got a message saying: `Your task failed to bootstrap on the Grid scheduler ....` What would you do and why?

1. Submit the task again.
2. Resubmit the task.
3. Submit a recovery task.

Show Hide answer.
`The task has not been submitted to the Grid, so there is nothing to resubmit (or to recover). One`

3.B) Imagine the following situation. You submitted the task and 50 jobs have failed in transferring the output files to the destination storage. You were told that there was a temporary issue with the storage and that it has been fixed now. What would you do and why?

1. Submit a completely new task from scratch.
2. Resubmit the failed jobs of the current task.
3. Submit a recovery task.

Show Hide answer.
```
Since the issue was a temporary problem not related to the jobs (which were otherwise finishing w
jobs would be the most reasonable choice.
```

3.C) Imagine the following situation. You submitted the task and 50 jobs were killed at the worker nodes, because they reached the default runtime limit. What would you do and why?

1. Submit a completely new task from scratch using a finer splitting.
2. Resubmit the failed jobs of the current task requesting a higher job runtime limit.
3. Submit a recovery task to analyze the failed jobs, but using a finer splitting.

Show Hide answer.
```
In this case the problem was with the jobs themselves. Resubmitting without changing the requeste
most probably fail again. Resubmitting requesting a higher job runtime may cause the jobs to be q
running. Submitting a new task would be a waste of resources as ~80% of the task has already comp
recovery task to analyze only the failed jobs, but it must be with a finer splitting.
```

3.D) In relation to 3.C, describe step-by-step the process to submit a recovery task in which the jobs are expected to use 1/4 the walltime than the original jobs. How many jobs will the recovery task have (approx.)? Will the output files from the recovery task be stored in the same directory as the output files from the original task? Will they be published in the same output dataset?

Help (which will not work after 26 July 2015): You can use the following existing task, `150626_112217:atanasi_crab_CRAB3_Advanced_Tutorial_June2015_Exercise3C`, which represents the exact situation described in 3.C. Doing `crab remake --task 150626_112217:atanasi_crab_CRAB3_Advanced_Tutorial_June2015_Exercise3C` you will create a CRAB project directory for this task, at which point you will be able to execute crab commands referring to this task. You can then submit your proposed recovery task. Will it publish in the same output dataset?

Exercise 4 - user script

The following exercises can be all run in one task if you want to save some time.

4.A) Run a task similar to the one in section Running CRAB to generate MC events of the CRAB3 (basic) tutorial, but wrapping `cmsRun` in a shell script. Don't forget to tell `cmsRun` to produce the `FrameworkJobReport.xml`. In the script, print some messages before `cmsRun` starts and after `cmsRun` finishes. Where are these messages printed?

Help:

Show Hide CMSSW parameter-set configuration file.
```
# Auto generated configuration file
# using:
# Revision: 1.19
# Source: /local/reps/CMSSW/CMSSW/Configuration/Applications/python/ConfigBuilder.py,v
# with command line options: MinBias_8TeV_cfi --conditions auto:startup -s GEN,SIM --datatier GEN
# --relval 9000,300 --eventcontent RAWSIM --io MinBias.io --python MinBias.py --no_exec --fileout

import FWCore.ParameterSet.Config as cms

process = cms.Process('SIM')

# Import of standard configurations
process.load('Configuration.StandardSequences.Services_cff')
```

```
process.load('SimGeneral.HepPDTESSource.pythiapdt_cfi')
process.load('FWCore.MessageService.MessageLogger_cfi')
process.load('Configuration.EventContent.EventContent_cff')
process.load('SimGeneral.MixingModule.mixNoPU_cfi')
process.load('Configuration.StandardSequences.GeometryRecoDB_cff')
process.load('Configuration.Geometry.GeometrySimDB_cff')
process.load('Configuration.StandardSequences.MagneticField_38T_cff')
process.load('Configuration.StandardSequences.Generator_cff')
process.load('IOMC.EventVertexGenerators.VtxSmearedRealistic8TeVCollision_cfi')
process.load('GeneratorInterface.Core.genFilterSummary_cff')
process.load('Configuration.StandardSequences.SimIdeal_cff')
process.load('Configuration.StandardSequences.EndOfProcess_cff')
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')

process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(10)
)

# Input source
process.source = cms.Source("EmptySource")

process.options = cms.untracked.PSet(

)

# Production Info
process.configurationMetadata = cms.untracked.PSet(
    version = cms.untracked.string('$Revision: 1.23 $'),
    annotation = cms.untracked.string('MinBias_8TeV_cfi nevts:10'),
    name = cms.untracked.string('Applications')
)

# Output definition
process.RAWSIMoutput = cms.OutputModule("PoolOutputModule",
    splitLevel = cms.untracked.int32(0),
    eventAutoFlushCompressedSize = cms.untracked.int32(5242880),
    outputCommands = process.RAWSIMEventContent.outputCommands,
    fileName = cms.untracked.string('minbias.root'),
    dataset = cms.untracked.PSet(
        filterName = cms.untracked.string(''),
        dataTier = cms.untracked.string('GEN-SIM')
    ),
    SelectEvents = cms.untracked.PSet(
        SelectEvents = cms.vstring('generation_step')
    )
)

# Additional output definition

# Other statements
process.genstepfilter.triggerConditions=cms.vstring("generation_step")
from Configuration.AlCa.GlobalTag import GlobalTag
process.GlobalTag = GlobalTag(process.GlobalTag, 'auto:startup', '')

process.generator = cms.EDFilter("Pythia6GeneratorFilter",
    pythiaPylistVerbosity = cms.untracked.int32(0),
    filterEfficiency = cms.untracked.double(1.0),
    pythiaHepMCVerbosity = cms.untracked.bool(False),
    comEnergy = cms.double(8000.0),
    maxEventsToPrint = cms.untracked.int32(0),
    PythiaParameters = cms.PSet(
        pythiaUESettings = cms.vstring('MSTU(21)=1     ! Check on possible errors during program
            'MSTJ(22)=2     ! Decay those unstable particles',
            'PARJ(71)=10 .  ! for which ctau  10 mm',
            'MSTP(33)=0     ! no K factors in hard cross sections',
            'MSTP(2)=1      ! which order running alphaS',
            'MSTP(51)=10042 ! structure function chosen (external PDF CTEQ6L1)',
```

```
            'MSTP(52)=2     ! work with LHAPDF',
            'PARP(82)=1.921 ! pt cutoff for multiparton interactions',
            'PARP(89)=1800. ! sqrts for which PARP82 is set',
            'PARP(90)=0.227 ! Multiple interactions: rescaling power',
            'MSTP(95)=6     ! CR (color reconnection parameters)',
            'PARP(77)=1.016 ! CR',
            'PARP(78)=0.538 ! CR',
            'PARP(80)=0.1   ! Prob. colored parton from BBR',
            'PARP(83)=0.356 ! Multiple interactions: matter distribution parameter',
            'PARP(84)=0.651 ! Multiple interactions: matter distribution parameter',
            'PARP(62)=1.025 ! ISR cutoff',
            'MSTP(91)=1     ! Gaussian primordial kT',
            'PARP(93)=10.0  ! primordial kT-max',
            'MSTP(81)=21    ! multiple parton interactions 1 is Pythia default',
            'MSTP(82)=4     ! Defines the multi-parton model'),
        processParameters = cms.vstring('MSEL=0         ! User defined processes',
            'MSUB(11)=1     ! Min bias process',
            'MSUB(12)=1     ! Min bias process',
            'MSUB(13)=1     ! Min bias process',
            'MSUB(28)=1     ! Min bias process',
            'MSUB(53)=1     ! Min bias process',
            'MSUB(68)=1     ! Min bias process',
            'MSUB(92)=1     ! Min bias process, single diffractive',
            'MSUB(93)=1     ! Min bias process, single diffractive',
            'MSUB(94)=1     ! Min bias process, double diffractive',
            'MSUB(95)=1     ! Min bias process'),
        parameterSets = cms.vstring('pythiaUESettings',
            'processParameters')
    )
)

# Path and EndPath definitions
process.generation_step = cms.Path(process.pgen)
process.simulation_step = cms.Path(process.psim)
process.genfiltersummary_step = cms.EndPath(process.genFilterSummary)
process.endjob_step = cms.EndPath(process.endOfProcess)
process.RAWSIMoutput_step = cms.EndPath(process.RAWSIMoutput)

# Schedule definition
process.schedule = cms.Schedule(
    process.generation_step,
    process.genfiltersummary_step,
    process.simulation_step,
    process.endjob_step,
    process.RAWSIMoutput_step
)

# Filter all path with the production filter sequence
for path in process.paths:
    getattr(process,path)._seq = process.generator * getattr(process,path)._seq
```

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_May2015_Exercise4A'

config.JobType.pluginName = 'PrivateMC'
config.JobType.psetName = 'pset_tutorial_MC_generation.py'
config.JobType.scriptExe = 'myscript.sh'

config.Data.outputPrimaryDataset = 'MinBias'
config.Data.splitting = 'EventBased'
config.Data.unitsPerJob = 10
config.Data.totalUnits = 30
config.Data.publication = True
```

```
config.Data.outputDatasetTag = config.General.requestName

config.Site.storageSite = <site where the user has permission to write>
```

Show Hide user script.

```
echo "================= CMSRUN starting ====================="
cmsRun -j FrameworkJobReport.xml -p PSet.py
echo "================= CMSRUN finished ====================="
```

4.B) Run a task as in 4.A, but save the messages into a text file. Consider this text file as an additional output file that should be transferred to the destination storage. Once transferred, retrieve them with `crab getoutput` and check that the messages are there.

Help:

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_May2015_Exercise4B'

config.JobType.pluginName = 'PrivateMC'
config.JobType.psetName = 'pset_tutorial_MC_generation.py'
config.JobType.scriptExe = 'myscript.sh'
config.JobType.outputFiles = ['output.txt']

config.Data.outputPrimaryDataset = 'MinBias'
config.Data.splitting = 'EventBased'
config.Data.unitsPerJob = 10
config.Data.totalUnits = 30
config.Data.publication = True
config.Data.outputDatasetTag = config.General.requestName

config.Site.storageSite = <site where the user has permission to write>
```

Show Hide user script.

```
echo "================= CMSRUN starting ====================" >> output.txt
cmsRun -j FrameworkJobReport.xml -p PSet.py
echo "================= CMSRUN finished ====================" >> output.txt
```

4.C) Run a task as in 4.A, but write the messages in a local text file, include this file in the input sandbox, and make your script read the messages from that file.

Help:

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_May2015_Exercise4B'

config.JobType.pluginName = 'PrivateMC'
config.JobType.psetName = 'pset_tutorial_MC_generation.py'
config.JobType.scriptExe = 'myscript.sh'
config.JobType.inputFiles = ['input.txt']

config.Data.outputPrimaryDataset = 'MinBias'
config.Data.splitting = 'EventBased'
config.Data.unitsPerJob = 10
config.Data.totalUnits = 30
config.Data.publication = True
config.Data.outputDatasetTag = config.General.requestName
```

```
config.Site.storageSite = <site where the user has permission to write>
```

Show Hide user script.

```
while read line
do
    if [ "$line" == "Before" ]; then
        continue
    elif [ "$line" == "After" ]; then
        cmsRun -j FrameworkJobReport.xml -p PSet.py
    else
        echo $line
    fi
done < input.txt
```

Show Hide local text file.

```
Before
================= CMSRUN starting ====================
After
================= CMSRUN finished ====================
```

4.D) Run a task as in 4.A, but pass the messages as arguments to your script.

Help:

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_May2015_Exercise4B'

config.JobType.pluginName = 'PrivateMC'
config.JobType.psetName = 'pset_tutorial_MC_generation.py'
config.JobType.scriptExe = 'myscript.sh'
# Arguments have to be in the form param=value, without white spaces, quotation marks nor additio
config.JobType.scriptArgs = ['Before=CMSRUN-starting', 'After=CMSRUN-finished']

config.Data.outputPrimaryDataset = 'MinBias'
config.Data.splitting = 'EventBased'
config.Data.unitsPerJob = 10
config.Data.totalUnits = 30
config.Data.publication = True
config.Data.outputDatasetTag = config.General.requestName

config.Site.storageSite = <site where the user has permission to write>
```

Show Hide user script.

```
beforemsg=""
aftermsg=""
for i in "$@"
do
case $i in
    Before=*)
    beforemsg="${i#*=}"
    ;;
    After=*)
    aftermsg="${i#*=}"
    ;;
esac
done
echo "================= $beforemsg ===================="
cmsRun -j FrameworkJobReport.xml -p PSet.py
echo "================= $aftermsg ===================="
```

4.E) Run a task as in 4.A, but defining the exit code of your script as 80500 (pass the exit code to CRAB; don't do `exit 80500` in the script). Once the task finishes (it should fail) check the status to see if the jobs are indeed reported as failed with exit code 80500.

Help:

Show Hide user script.

```
echo "================= CMSRUN starting ====================="
cmsRun -j FrameworkJobReport.xml -p PSet.py
echo "================= CMSRUN finished ====================="

exitCode=80500
exitMessage="This is a test to see if I can pass exit code 80500 to CRAB."
errorType=""

if [ -e FrameworkJobReport.xml ]
then
    cat << EOF > FrameworkJobReport.xml.tmp
<FrameworkJobReport>
<FrameworkError ExitStatus="$exitCode" Type="$errorType" >
$exitMessage
</FrameworkError>
EOF
    tail -n+2 FrameworkJobReport.xml >> FrameworkJobReport.xml.tmp
    mv FrameworkJobReport.xml.tmp FrameworkJobReport.xml
else
    cat << EOF > FrameworkJobReport.xml
<FrameworkJobReport>
<FrameworkError ExitStatus="$exitCode" Type="$errorType" >
$exitMessage
</FrameworkError>
</FrameworkJobReport>
EOF
fi
```

Exercise 5 - LHE

5.A) The objective of this exercise it to run a MC generation on LHE files. Using the Running MC generation on LHE files twiki as a pointer, prepare (but not submit yet!) a CRAB3 configuration to run on the Grid using the pset and the LHE file you can find here `/afs/cern.ch/cms/ccs/wm/scripts/Crab/CRAB3/AdvTutorial/Exercise_5/`. Use `config.JobType.inputFiles` to pass the LHE file to the jobs. The target is to have 10 jobs that will run for 8 hours. Run your job locally on 1000 events before submitting it with CRAB (should take less than a minute), and use the `Timing` service of CMSSW to get a time per event estimation. Everything should use CMSSW_5_3_22. Publication should be enabled in your configuration.

Help:

Show Hide Adding Timing service to the PSet.

```
##### These are the lines you can add to your pset get the estimates in the FrameworkJobReport fi
process.Timing = cms.Service("Timing",
    summaryOnly = cms.untracked.bool(True)
)

##### For your information CRAB3 also add the following two lines in addition to the previous thr
#process.CPU = cms.Service("CPU")

#process.SimpleMemoryCheck = cms.Service("SimpleMemoryCheck")
```

Show Hide Runing cmsRun locally to estimate the time per event.

```
cmsRun -j FrameworkJobReport.xml -p pset_MC_generation_LHE.py
```

```
You can open the FrameworkJobReport.xml file now and look for the AvgEventTime, which is what CRA
```

Show Hide CRAB configuration file.

```
from CRABClient.UserUtilities import config
config = config()

config.General.requestName = 'CRAB3_Advanced_Tutorial_May2015_Exercise5A'

config.JobType.pluginName = 'PrivateMC'
config.JobType.generator = 'lhe'
config.JobType.psetName = 'pset_MC_generation_LHE.py'
config.JobType.inputFiles = ['dynlo.lhe']

config.Data.outputPrimaryDataset = 'TutorialMay2015Exercise5A'
config.Data.splitting = 'EventBased'
JOB_WALLTIME = 8*3600
TIME_PER_EVENT = 0.25
config.Data.unitsPerJob = int(JOB_WALLTIME / TIME_PER_EVENT)
NJOBS = 10
config.Data.totalUnits = config.Data.unitsPerJob * NJOBS
config.Data.publication = True
config.Data.outputDatasetTag = 'MC_generation_LHE'

config.Site.storageSite = <storage-site>
```

5.B) Try to submit the task. Is it there a problem? Try to explain what is going on.

Help:

Show Hide answer.

```
You are probably getting the following error, because passing the LHE file directly in the input
allowed limit of 100Mb:

Will use CRAB configuration file /afs/cern.ch/user/m/mmascher/tutorial/Exercise_5/crabConfig.py

Error contacting the server.
Server answered with: Invalid input parameter
Reason is: File is bigger then allowed limit of 10485760B
```

5.C) Copy the dynlo.lhe file to your destination storage under `/store/user/<your-username>/dynlo.lhe` and try to submit the task again removing the `JobType.inputFiles` parameter and accessing the file locally. Remember to modify the LHEInputSource and whitelist your site in the CRAB configuration so that your jobs will only run there.

Help:

Show Hide Get the PFN for the LHEInputSource

```
/afs/cern.ch/user/b/belforte/public/mybin/lfn2pfn.sh <site> [<LFN>]
```

Show Hide CRAB configuration file

```
#config.JobType.inputFiles = ['dynlo.lhe']
config.Site.whitelist = ['T2_CH_CERN']
```

Show Hide PSet configuration file

```
process.source = cms.Source("LHESource",
    fileNames = cms.untracked.vstring('gsi
ftp://eoscmsftp.cern.ch//eos/cms/store/user/<username>/dynlo.lhe ')
)
```

5.D) Remove the whitelist and have the jobs run all the sites of your storage element nation. For example, if you are storing files to `T2_IT_Legnaro` you are going to run jobs only on italian sites (`T2_IT*`). Can you

individuate the differences between the logfiles between 5C and this exercise ?

Help:

Show Hide CRAB configuration file

```
#config.JobType.inputFiles = ['dynlo.lhe']
config.Site.whitelist = ['T2_CH*']
```

Exercise 6 - CRABAPI library

6) Using the same CMSSW parameter-set configuration as in exercise 2, submit four identical tasks to analyze the following four MC input datasets:

- `/DoubleMuParked/Run2012A-22Jan2013-v1/AOD`
- `/DoubleMuParked/Run2012B-22Jan2013-v1/AOD`
- `/DoubleMuParked/Run2012C-22Jan2013-v1/AOD`
- `/DoubleMuParked/Run2012D-22Jan2013-v1/AOD`

Don't do `crab submit` four times; instead use the `crabCommand` API from the CRABAPI library in a script. There is no need to analyze the whole datasets; just a few lumis or files is enough. Use the CRABAPI library to check the status, resubmit failed jobs, get the report and retrieve the output files, from the tasks you submitted.

Help:

Show Hide suggestion.

```
Create a python script named "multicrab" (with permissions 744) which you should be able to run i
     ./multicrab --crabCmd CMD [--workArea WAD --crabCmdOpts OPTS]
where CMD is the crab command, WAD is a work area directory with many CRAB project directories in
command.
```

Show Hide multicrab script skeleton.

```
#!/usr/bin/env python
"""
This is a small script that does the equivalent of multicrab.
"""
import os
from optparse import OptionParser


def getOptions():
    """
    Parse and return the arguments provided by the user.
    """
    usage = ("Usage: %prog --crabCmd CMD [--workArea WAD --crabCmdOpts OPTS]"
             "\nThe multicrab command executes 'crab CMD OPTS' for each project directory contain
             "\nUse multicrab -h for help")

    parser = OptionParser(usage=usage)

    parser.add_option('-c', '--crabCmd',
                      dest = 'crabCmd',
                      default = '',
                      help = "crab command",
                      metavar = 'CMD')

    parser.add_option('-w', '--workArea',
                      dest = 'workArea',
                      default = '',
                      help = "work area directory (only if CMD != 'submit')",
                      metavar = 'WAD')
```

```
    parser.add_option('-o', '--crabCmdOpts',
                      dest = 'crabCmdOpts',
                      default = '',
                      help = "options for crab command CMD",
                      metavar = 'OPTS')

    (options, arguments) = parser.parse_args()

    if arguments:
        parser.error("Found positional argument(s): %s." % (arguments))
    if not options.crabCmd:
        parser.error("(-c CMD, --crabCmd=CMD) option not provided.")
    if options.crabCmd != 'submit':
        if not options.workArea:
            parser.error("(-w WAR, --workArea=WAR) option not provided.")
        if not os.path.isdir(options.workArea):
            parser.error("'%s' is not a valid directory." % (options.workArea))

    return options


def main():

    options = getOptions()

    # Do something.


if __name__ == '__main__':
    main()
```

Show Hide multicrab script.

```
#!/usr/bin/env python
"""
This is a small script that does the equivalent of multicrab.
"""
import os
from optparse import OptionParser

from CRABAPI.RawCommand import crabCommand
from CRABClient.ClientExceptions import ClientException
from httplib import HTTPException


def getOptions():
    """
    Parse and return the arguments provided by the user.
    """
    usage = ("Usage: %prog --crabCmd CMD [--workArea WAD --crabCmdOpts OPTS]"
             "\nThe multicrab command executes 'crab CMD OPTS' for each project directory contain
             "\nUse multicrab -h for help")

    parser = OptionParser(usage=usage)

    parser.add_option('-c', '--crabCmd',
                      dest = 'crabCmd',
                      default = '',
                      help = "crab command",
                      metavar = 'CMD')

    parser.add_option('-w', '--workArea',
                      dest = 'workArea',
                      default = '',
                      help = "work area directory (only if CMD != 'submit')",
                      metavar = 'WAD')
```

```
    parser.add_option('-o', '--crabCmdOpts',
                      dest = 'crabCmdOpts',
                      default = '',
                      help = "options for crab command CMD",
                      metavar = 'OPTS')

    (options, arguments) = parser.parse_args()

    if arguments:
        parser.error("Found positional argument(s): %s." % (arguments))
    if not options.crabCmd:
        parser.error("(-c CMD, --crabCmd=CMD) option not provided.")
    if options.crabCmd != 'submit':
        if not options.workArea:
            parser.error("(-w WAR, --workArea=WAR) option not provided.")
        if not os.path.isdir(options.workArea):
            parser.error("'%s' is not a valid directory." % (options.workArea))

    return options


def main():

    options = getOptions()

    # The submit command needs special treatment.
    if options.crabCmd == 'submit':

        #----------------------------------------------------------
        # This is the base config:
        #----------------------------------------------------------
        from CRABClient.UserUtilities import config
        config = config()

        config.General.requestName = None
        config.General.workArea = 'CRAB3_Advanced_Tutorial_May2015_Exercise6'

        config.JobType.pluginName = 'Analysis'
        config.JobType.psetName = 'pset_tutorial_analysis.py'

        config.Data.inputDataset = None
        config.Data.splitting = 'LumiBased'
        config.Data.unitsPerJob = 10
        config.Data.totalUnits = 30
        config.Data.outputDatasetTag = None

        config.Site.storageSite = None # Choose your site.
        #----------------------------------------------------------

        # Will submit one task for each of these input datasets.
        inputDatasets = [
                         '/DoubleMuParked/Run2012A-22Jan2013-v1/AOD',
                         '/DoubleMuParked/Run2012B-22Jan2013-v1/AOD',
                         '/DoubleMuParked/Run2012C-22Jan2013-v1/AOD',
                         '/DoubleMuParked/Run2012D-22Jan2013-v1/AOD',
                        ]

        for inDS in inputDatasets:
            # inDS is of the form /A/B/C. Since B is unique for each inDS, use this in the CRAB r
            config.General.requestName = inDS.split('/')[2]
            config.Data.inputDataset = inDS
            config.Data.outputDatasetTag = '%s_%s' % (config.General.workArea, config.General.req
            # Submit.
            try:
                print "Submitting for input dataset %s" % (inDS)
                crabCommand(options.crabCmd, config = config, *options.crabCmdOpts.split())
            except HTTPException as hte:
```

```
                print "Submission for input dataset %s failed: %s" % (inDS, hte.headers)
            except ClientException as cle:
                print "Submission for input dataset %s failed: %s" % (inDS, cle)

    # All other commands can be simply executed.
    elif options.workArea:

        for dir in os.listdir(options.workArea):
            projDir = os.path.join(options.workArea, dir)
            if not os.path.isdir(projDir):
                continue
            # Execute the crab command.
            msg = "Executing (the equivalent of): crab %s --dir %s %s" % (options.crabCmd, projDi
            print "-"*len(msg)
            print msg
            print "-"*len(msg)
            try:
                crabCommand(options.crabCmd, dir = projDir, *options.crabCmdOpts.split())
            except HTTPException as hte:
                print "Failed executing command %s for task %s: %s" % (options.crabCmd, projDir,
            except ClientException as cle:
                print "Failed executing command %s for task %s: %s" % (options.crabCmd, projDir,


if __name__ == '__main__':
    main()
```

-- AndresTanasijczuk - 2015-05-19

This topic: CMSPublic > CRAB3AdvancedTutorial
Topic revision: r23 - 2016-05-02 - StefanoBelforte