Table of Contents

CRAB3 Cheat Sheet

Complete:	Go to SWGuideCrab

CRAB Environment setup

Like all CMS software, you need to first start by configuring your environment to see CRAB3. As of 2020 CRAB Client is distributed as an external inside CMSSW, therefore.. if you have CMSSW, you also have CRAB ! The recommended way to install CMSSW at sites is via CVMFS, and here we assume that this holds for your site as well, definitely it is true for large, commonly used, plaftorms like lxplus@CERN and lpc@FNAL

Since CRAB comes via CMSSW, `crab` command is already defined as soon as your CMS environment is defined, same as commands like `scram`, `cmsrel` or `dasgoclient`

Indeed you can type `crab` as soon as you log on e.g. lxplus. On the other hand CRAB has a few software dependencies which are not satisfied by the bare operating system, so the former action will fail and will tell you to do

```
cmsenv
```

(of course when the current work directory is a valid CMSSW tree created via `cmsrel`) and CRAB will work after that.

Show Hide Example.

```
* ********************************************************************
* Welcome to lxplus785.cern.ch, CentOS, 7.8.2003
* Archive of news is available in /etc/motd-archive
* Reminder: you have agreed to the CERN
*   computing rules, in particular OC5. CERN implements
*   the measures necessary to ensure compliance.
*   https://cern.ch/ComputingRules
* Puppet environment: production, Roger state: production
* Foreman hostgroup: lxplus/nodes/login
* Availability zone: cern-geneva-c
* LXPLUS Public Login Service - http://lxplusdoc.web.cern.ch/
* A C8 based lxplus8.cern.ch is now available
* Tuesday November 24th lxplus6.cern.ch will terminate - http://cern.ch/go/j9cD
* ********************************************************************
belforte@lxplus785/~> crab

Error: missing SSL support in pycurl. Make sure you do cmsenv first. Exiting...
pycurl version is: libcurl/7.29.0 NSS/3.44 zlib/1.2.7 libidn/1.28 libssh2/1.8.0

belforte@lxplus785/~> cd WORK/CMSSW/SL7/CMSSW_10_6_12/
belforte@lxplus785/CMSSW_10_6_12> cmsenv
belforte@lxplus785/CMSSW_10_6_12> crab
You have not specified a command.

Valid commands are:
  checkusername
  checkwrite (chk)
  getlog (log)
  getoutput (output) (out)
  kill
  preparelocal
  proceed
  purge
  remake (rmk)
  report (rep)
```

```
  resubmit
  status (st)
  submit (sub)
  tasks
  uploadlog (uplog)
To get single command help run:
  crab command --help|-h

For more information on how to run CRAB-3 please follow this link:
https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCRAB3Tutorial
```

The above minimal setup (just `cmsenv`) is sufficient to execute CRAB commands interactively, but if you want to use CRBA Python API to execute commands via python calls in a script of yours you will need to do (replace `sh` with `csh` if needed for you shell)

```
source /cvmfs/cms.cern.ch/common/crab-setup.sh
```

To know more

CMSCrabClient twiki page has more about CRAB Client: setup, available variants, API use, debug and development contribution

Grid Setup

At many sites (lxplus@CERN e.g.), the grid UI already comes with the environment (if you can run grid-proxy-init, then your environment already has the grid UI). Otherwise, you'll need to also source a grid UI. Different sites have their UIs stored in different locations (ask your admins!)

Since CRAB will run your jons on the Grid, you need to have a valid Grid credential when using. You should execute this command every time you log in on a machine

```
voms-proxy-init -voms cms -rfc  -valid 192:00
```

CRAB3 configurations

CRAB3 configurations are written in python (compared with CRAB2s ini-based format). This allows powerful and flexible configurations. Please also note that CRAB3 configuration files must end in ".py" to be properly imported.

Sample CRAB configurations

If you would like to start from scratch, the following example configurations are sufficient to get started. Make sure to replace the colored sections with your own values

Show Hide Data processing config.

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

config.General.requestName = 'test1'

config.JobType.pluginName = 'Analysis'
# Name of the CMSSW configuration file
config.JobType.psetName = 'pset.py'

config.Data.inputDataset = '/GenericTTbar/HC-CMSSW_5_3_1_START53_V5-v1/GEN-SIM-RECO'
config.Data.splitting = 'LumiBased'
config.Data.unitsPerJob = 100
```

```
config.Data.publication = True
# This string is used to construct the output dataset name
config.Data.outputDatasetTag = 'CRAB3_Analysis_test1'

# These values only make sense for processing data
#    Select input data based on a lumi mask
config.Data.lumiMask = 'Cert_190456-208686_8TeV_PromptReco_Collisions12_JSON.txt'
#    Select input data based on run-ranges
config.Data.runRange = '190456-194076'

# Where the output files will be transmitted to
config.Site.storageSite = 'T2_US_Nowhere'
```

Show Hide private Monte Carlo generation config.

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

config.General.requestName   = 'test1'

config.JobType.pluginName = 'PrivateMC'
# Name of the CMSSW configuration file
config.JobType.psetName = 'pset.py'

# This string determines the primary dataset of the newly-produced outputs.
# For instance, this dataset will be named /CrabTestSingleMu/something/USER
config.Data.outputPrimaryDataset = 'CrabTestSingleMu'
config.Data.splitting = 'EventBased'
config.Data.unitsPerJob = 100
config.Data.totalUnits = 1000
config.Data.publication = True

# This string is used to construct the output dataset name
config.Data.outputDatasetTag = 'CRAB3_MC_generation_test1'

# Where the output files will be transmitted to
config.Site.storageSite = 'T2_US_Nowhere'
```

Run CRAB3

Once you have a CRAB3 configuration file and a valid CMSSW configuration file, you're ready to run CRAB3! These examples assume that you've named your CRAB configuration file `crabConfig.py` and left the default value for `config.General.requestName`.

- Submit the jobs:

```
crab submit --config=crabConfig.py
```

- Check the job status:

```
crab status --dir=crab_test1
```

- Retrieve the logs from all completed jobs:

```
crab getlog --dir=crab_test1
```

- See the list of good lumis for your task:

```
crab report --dir=crab_test1
```

Similar to CRAB2's behavior, submitting a CRAB3 task creates a directory (we call it CRAB project directory) storing all the information about the request. The `--dir/-d` option for the other commands accepts

this directory as an input.

More information

For a more detailed manual, please see the CRAB3 Tutorial or consult the available documentation about CRAB3 which is all linked from the Software Guide on CRAB page.

Sample CMSSW configurations

If you somehow manage to not have any CMSSW configuration sitting around, you can use these to test CRAB3. They are known to work with CMSSW_5_3_4

Show Hide Data processing CMSSW config.

```
import FWCore.ParameterSet.Config as cms
process = cms.Process('NoSplit')
process.source = cms.Source('PoolSource',
  fileNames = cms.untracked.vstring("/store/mc/HC/GenericTTbar/GEN-SIM-RECO/CMSSW_5_3_1_START53_V
  skipEvents = cms.untracked.uint32(0),
)

process.dump = cms.EDAnalyzer("EventContentAnalyzer", listContent=cms.untracked.bool(False), getD
process.load("FWCore.MessageService.MessageLogger_cfi")
process.MessageLogger.cerr.FwkReport.reportEvery = 10

process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(50)
)

process.Timing = cms.Service("Timing",
    useJobReport = cms.untracked.bool(True),
    summaryOnly = cms.untracked.bool(True),
)

process.o = cms.OutputModule("PoolOutputModule", fileName = cms.untracked.string("dumper.root"),

process.out = cms.EndPath(process.o)

process.p = cms.Path(process.dump)
```

Show Hide private Monte Carlo generation CMSSW config.

```
# Auto generated configuration file
# using:
# Revision: 1.381.2.11
# Source: /local/reps/CMSSW/CMSSW/Configuration/PyReleaseValidation/python/ConfigBuilder.py,v
# with command line options: TTbar_Tauola_7TeV_cfi.py -s GEN,FASTSIM,HLT:GRun --conditions=auto:s
import FWCore.ParameterSet.Config as cms

process = cms.Process('HLT')

# import of standard configurations
process.load('Configuration.StandardSequences.Services_cff')
process.load('SimGeneral.HepPDTESSource.pythiapdt_cfi')
process.load('FWCore.MessageService.MessageLogger_cfi')
process.load('FastSimulation.Configuration.EventContent_cff')
process.load('FastSimulation.PileUpProducer.PileUpSimulator_NoPileUp_cff')
process.load('FastSimulation.Configuration.Geometries_START_cff')
process.load('Configuration.StandardSequences.MagneticField_38T_cff')
process.load('Configuration.StandardSequences.Generator_cff')
process.load('GeneratorInterface.Core.genFilterSummary_cff')
process.load('FastSimulation.Configuration.FamosSequences_cff')
process.load('IOMC.EventVertexGenerators.VtxSmearedParameters_cfi')
process.load('HLTrigger.Configuration.HLT_GRun_Famos_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: 1.10 $'),
    annotation = cms.untracked.string('TTbar_Tauola_7TeV_cfi.py nevts:10'),
    name = cms.untracked.string('PyReleaseValidation')
)

# Output definition

process.AODSIMoutput = cms.OutputModule("PoolOutputModule",
    eventAutoFlushCompressedSize = cms.untracked.int32(15728640),
    outputCommands = process.AODSIMEventContent.outputCommands,
    fileName = cms.untracked.string('MyTTBarTauolaTest.root'),
    dataset = cms.untracked.PSet(
        filterName = cms.untracked.string(''),
        dataTier = cms.untracked.string('AODSIM')
    ),
    SelectEvents = cms.untracked.PSet(
        SelectEvents = cms.vstring('generation_step')
    )
)

# Additional output definition

# Other statements
process.genstepfilter.triggerConditions=cms.vstring("generation_step")
process.famosSimHits.SimulateCalorimetry = True
process.famosSimHits.SimulateTracking = True
process.simulation = cms.Sequence(process.simulationWithFamos)
process.HLTEndSequence = cms.Sequence(process.reconstructionWithFamos)
process.Realistic8TeVCollisionVtxSmearingParameters.type = cms.string("BetaFunc")
process.famosSimHits.VertexGenerator = process.Realistic8TeVCollisionVtxSmearingParameters
process.famosPileUp.VertexGenerator = process.Realistic8TeVCollisionVtxSmearingParameters
from Configuration.AlCa.GlobalTag import GlobalTag
process.GlobalTag = GlobalTag(process.GlobalTag, 'auto:startup_GRun', '')

process.generator = cms.EDFilter("Pythia6GeneratorFilter",
    ExternalDecays = cms.PSet(
        Tauola = cms.untracked.PSet(
            UseTauolaPolarization = cms.bool(True),
            InputCards = cms.PSet(
                mdtau = cms.int32(0),
                pjak2 = cms.int32(0),
                pjak1 = cms.int32(0)
            )
        ),
        parameterSets = cms.vstring('Tauola')
    ),
    pythiaPylistVerbosity = cms.untracked.int32(0),
    filterEfficiency = cms.untracked.double(1.0),
    pythiaHepMCVerbosity = cms.untracked.bool(False),
    comEnergy = cms.double(7000.0),
    maxEventsToPrint = cms.untracked.int32(0),
    PythiaParameters = cms.PSet(
```

```
        pythiaUESettings = cms.vstring('MSTJ(11)=3     ! Choice of the fragmentation function',
            'MSTJ(22)=2     ! Decay those unstable particles',
            'PARJ(71)=10 .  ! for which ctau  10 mm',
            'MSTP(2)=1      ! which order running alphaS',
            'MSTP(33)=0     ! no K factors in hard cross sections',
            'MSTP(51)=10042 ! structure function chosen (external PDF CTEQ6L1)',
            'MSTP(52)=2     ! work with LHAPDF',
            'MSTP(81)=1     ! multiple parton interactions 1 is Pythia default',
            'MSTP(82)=4     ! Defines the multi-parton model',
            'MSTU(21)=1     ! Check on possible errors during program execution',
            'PARP(82)=1.8387   ! pt cutoff for multiparton interactions',
            'PARP(89)=1960. ! sqrts for which PARP82 is set',
            'PARP(83)=0.5   ! Multiple interactions: matter distrbn parameter',
            'PARP(84)=0.4   ! Multiple interactions: matter distribution parameter',
            'PARP(90)=0.16  ! Multiple interactions: rescaling power',
            'PARP(67)=2.5    ! amount of initial-state radiation',
            'PARP(85)=1.0  ! gluon prod. mechanism in MI',
            'PARP(86)=1.0  ! gluon prod. mechanism in MI',
            'PARP(62)=1.25   ! ',
            'PARP(64)=0.2    ! ',
            'MSTP(91)=1      !',
            'PARP(91)=2.1   ! kt distribution',
            'PARP(93)=15.0  ! '),
        processParameters = cms.vstring('MSEL      = 0     ! User defined processes',
            'MSUB(81)  = 1     ! qqbar to QQbar',
            'MSUB(82)  = 1     ! gg to QQbar',
            'MSTP(7)   = 6     ! flavour = top',
            'PMAS(6,1) = 175.  ! top quark mass'),
        parameterSets = cms.vstring('pythiaUESettings',
            'processParameters')
    )
)


# Path and EndPath definitions
process.generation_step = cms.Path(process.pgen_genonly)
process.reconstruction = cms.Path(process.reconstructionWithFamos)
process.genfiltersummary_step = cms.EndPath(process.genFilterSummary)
process.AODSIMoutput_step = cms.EndPath(process.AODSIMoutput)

# Schedule definition
process.schedule = cms.Schedule(process.generation_step,process.genfiltersummary_step)
process.schedule.extend(process.HLTSchedule)
process.schedule.extend([process.reconstruction,process.AODSIMoutput_step])
# filter all path with the production filter sequence
for path in process.paths:
   getattr(process,path)._seq = process.generator * getattr(process,path)._seq

# customisation of the process.

# Automatic addition of the customisation function from HLTrigger.Configuration.customizeHLTforMC
from HLTrigger.Configuration.customizeHLTforMC import customizeHLTforMC

#call to customisation function customizeHLTforMC imported from HLTrigger.Configuration.customize
process = customizeHLTforMC(process)

# End of customisation functions
```

--++ Conversion from Crab2 Crab2 is so old as of this writing that not only is not supported anymore, but there is not reason to expect that following stuff is needed or working. Anyhow, for completeness, here it is:

A CRAB2 to CRAB3 conversion script exists, which will usually produce a working CRAB3 configuration, removing obsolete parameters and warning about incompatible parameters. This script is named `crab2cfgTOcrab3py` and should be run as

```
crab2cfgTOcrab3py [crab2configName.cfg] [crab3configName.py]
```

It can be executed after sourcing the CRAB3 environment.

-- AndresTanasijczuk - 07 Oct 2014

This topic: CMSPublic > CRAB3CheatSheet
Topic revision: r10 - 2020-09-23 - StefanoBelforte