Table of Contents

Deployment of CRAB REST Interface on SLC7 machines

Complete:	Go to SWGuideCrab

Introduction

This twiki explains how to deploy the CRAB server frontend (a.k.a. the CRAB REST Interface). It will guide you through the steps required to:

1. Ask for an Oracle account and install the CRAB database schema from scratch.
2. Get a virtual machine (VM) with the right architecture from the CERN OpenStack Cloud Infrastructure.
3. Install the required software on the machine.
4. Configure the machine.

Note: Legend of colors for the examples:

```
Commands to execute

Output sample of the executed commands

Configuration files

Other files
```

Note: We will sometimes use the short cut "REST Interface" or just "REST" to refer to "CRAB REST Interface".

Get a CERN Oracle database account

Go to https://resources.web.cern.ch/resources/Manage/Oracle/Subscribe.aspx and subscribe to the "oracle-general-purpose-users" e-group.

Note: You will need to wait for approval before you are actually added to the e-group. You will get an e-mail when that happens. After you have subscribed, if you click on "oracle-general-purpose-users" from the subscription page above, you will see a button with the label "Remove me from members". But this doesn t mean that you have been approved. To see your status, you can click again on "oracle-general-purpose-users" and go to the "Members" tab. You will only be able to see your name, and until you have been approved in the far-right column it will say "Waiting for approval".

Once you have been approved, go to https://resources.web.cern.ch/resources/Manage/Oracle/NewOracleAccount.aspx and fill out a request to get a CERN Oracle database account:

```
Login: crab3_<username>
Database: devdb11
Description: For private CRAB3 server instance.
```

If you want to be able to see/edit your CERN Oracle database, install Oracle SQL developer (requires to create an Oracle account). Download this file and put it in `/etc/` (at least for linux, for other OSes the locations in which SQL Developer looks for the `tnsnames.ora` file may be different). When creating the connection to the CERN Oracle DB in SQL developer, input an arbitrary connection name, username (same as

the "Login" you chose for the account, something like `crab3_username`) and your chosen password. In the "Connection type" dropdown, select "TNS". Finally, in the "Network Alias" dropdown, select the database instance (`devdb11` in our case). Connection should be now set up, so click the "Test" button to see if it works, and the "Save" button to store the connection details for later use.

Get and install a virtual machine

The CRAB REST Interface uses the code and the infrastructure provided by the HTTP group. The production and pre-production CRAB REST Interfaces are deployed on the CMSWEB cluster. The official tutorial of the HTTP group for installing your own version of the CRAB REST Interface on a VM can be found in https://cms-http-group.web.cern.ch/cms-http-group/tutorials/environ/vm-setup.html, in particular section "1.1.1. CERN virtual machine". That's the tutorial we suggest to follow, up to point "6. Set up authentication" inclusively, to prepare your VM. The only suggested change is to use the following parameters when requesting the VM:

```
Availability Zone: Any Availability Zone
Instance Name: <any name you want>
Flavor: m1.large
Instance Count: 1
Instance Boot Source: Boot from image
Image Name: CC7 - x86_64 (the latest)
```

Note: For the interested reader, there is a CERN OpenStack Private Cloud Guide at http://clouddocs.web.cern.ch/clouddocs/

Machine preparation

If, as suggested in section Get and install a virtual machine, you followed the HTTP group tutorial up to point "6. Set up authentication" inclusively (the recommended way!), then you don't need to perform the following steps and you can go directly to section REST Interface installation and configuration. But check first Install voms clients.

The `Deploy` script will request a Grid host certificate for your VM and put it in the right directory with the right permissions. This is done in an *ad-hoc* way (the script does exactly the same as what you would do manually; there is no API). ~~If the `Deploy` script fails when running `careq.cern.ch`, then you can request the Grid host certificate manually as described below and then re-run `cfg/Deploy -t dummy -s post $PWD system/devvm`.~~

Requesting a Grid host certificate for your VM

a) Using a browser that has your Grid user certificate imported on it, go to https://gridca.cern.ch/.

- Click on "New Grid Host Certificate".
- Click on "Request certificate using OpenSSL (recommended for Linux machines)".
- Choose the VM host name from the pull-down menu under "Certificate Subject".
- Click "Proceed to request submission".

~~b) Now you must have got the instructions on how to prepare the Grid host certificate request.~~

- Log in to your VM.
- Make sure you have the `~/.globus` directory with your valid Grid user certificate (files `usercert.pem` and `userkey.pem`).
- If your Grid user certificate was issued by a certification authority other than CERN CA, associate your CERN primary account to the certificate. You can do that in https://gridca.cern.ch/gridca/.

- Source the UI (`source /afs/cern.ch/cms/LCG/LCG-2/UI/cms_ui_env.sh`) if not done by default.
- Run the following command (for example in the `private` directory of your home area):

```
cd ~/private
openssl req -new -subj "/CN=`hostname -f`" -out newcsr.csr -nodes -sha512 -newkey rsa:2048
```

- Two files should have been created, `privkey.pem` and `newcsr.csr`. The first contains the private key and the second the certificate request that you should send to CERN CA. Open the file `newcsr.csr`, copy all its content and paste it in the webpage in the field "Certificate Request".
- Click "Submit".

c) Download the certificate by clicking on "Base 64 encoded" under "Download Certificate". (The default name of the downloaded file is `host.cert`.)

d) Copy the `host.cert` file to your VM (to the same directory where you have the `privkey.pem` file, for example the `private` directory of your home area).

```
scp host.cert <username>@lxplus.cern.ch:~/private/
```

e) *Optional*: Create a certificate in pkcs12 format.

In your VM run the following command:

```
openssl pkcs12 -export -inkey privkey.pem -in host.cert -out myCertificate.p12
```

?) Follow the instructions given on gridca page up to a point until you generate myCertificate.p12 file. Do not delete any of the files.

f) Move the certificate and private key to the directory `/etc/grid-security/`, change the ownership of the files to `root:root` and protect the private key so that only `root` can read it. Remove the file `newcsr.csr`.

```
cd ~/private/
sudo mkdir /etc/grid-security
sudo mv host.cert /etc/grid-security/hostcert.pem
sudo mv privkey.pem /etc/grid-security/hostkey.pem
sudo chmod 400 /etc/grid-security/hostkey.pem
sudo chown root:root /etc/grid-security/host{cert,key}.pem
rm newcsr.csr
```

1) Install git, clone the `dmwm/deployment` repository and use the `Deploy` script for preparing the VM.

~~**Note from the HTTP group tutorial**: The Deploy command below will also check if it is possible to resize the current root partition on the fly, asking for your confirmation for doing so or skip. This is recommended, because the default SLC5/6 installations do not size partitions to use the entire allocated disk. It is also useful if you later resize the VM to a different flavor (i.e. with more disk space). Note, however, the partition resizing procedure is not very reliable, so do never, ever, run it on a production server VM, nor expect it to work for VM images other than the SLC5/6 ones.~~

Does it still do that?

```
sudo yum -y install git.x86_64
mkdir -p /tmp/foo
cd /tmp/foo
git clone git://github.com/dmwm/deployment.git cfg
sudo -l
cfg/Deploy -t dummy -s post $PWD system/devvm
```

```
INFO: can resize '/' to increase in 75162 MB. Proceed? (y/n)

n
```

Clean-up:

```
less /tmp/foo/.deploy/* # if you want to check what happened
cd ~
rm -rf /tmp/foo
```

2) Request proxy renewal rights for your VM.

Send an e-mail to px.support(AT)cern.ch with Cc to cms-service-webtools(AT)cern.ch.

E-mail subject:

```
myproxy registration request for <hostname>
```

E-mail body:

```
Could you please add the following host certificate to myproxy.cern.ch trusted retrievers, author
This is a development server for CM web services and requires use of grid proxy certificates.

/DC=ch/DC=cern/OU=computers/CN=<hostname>

Regards,
<Your Name>
```

where `<hostname>` is the full host name as shown by the command `hostname -f`. Run `openssl x509 -in /etc/grid-security/hostcert.pem -noout -subject` and verify that the host certificate is correct in the message body.

3) Log out.

Note from the HTTP group tutorial: It is also recommended to reboot the server, specially if you have resized partitions on the fly, but it isn t really necessary. If you use SSH persistent connections, be sure to terminate the master connection before logging in again. The commands in step 1) modify users and groups, and it s important they apply when you execute the next steps. This requires completely new log-in, and using persistent SSH connections may cause you to use cached credentials.

4) Log in.

SSH into your VM again and run the following to double check that you got the multiple service accounts:

```
id # should print out large number local _foo groups now
```

Install voms clients

Seems to come preinstalled in CC7 ~~If voms-proxy-init is not available in the VM, install the voms clients. Check which version should be installed by running~~ ~~`yum provides`~~ ~~(in the example below it tells to install voms-clients-2.0.12-1.el6.x86_64).~~

```
sudo yum provides */voms-proxy-init

Loaded plugins: changelog, kernel-module, priorities, protectbase, security, tsflags, versionlock
147 packages excluded due to repository priority protections
0 packages excluded due to repository protections
EGI-trustanchors/filelists
```

```
glite-SCAS/filelists
glite-SCAS_ext/filelists
glite-SCAS_updates/filelists
slc6-extras/filelists_db
slc6-updates/filelists_db
voms-clients-2.0.12-1.el6.x86_64 : Virtual Organization Membership Service Clients
Repo        : epel
Matched from:
Filename    : /usr/bin/voms-proxy-info
```

Install the voms-client version specified in the output of yum provides

```
sudo yum install voms-clients-2.0.12-1.el6.x86_64
```

You may also need to copy the directories `/etc/vomses` and `/etc/grid-security/vomsdir/cms` from lxplus.

```
sudo scp -r <username>@lxplus.cern.ch:/etc/vomses /etc/vomses
sudo mkdir /etc/grid-security/vomsdir
sudo scp -r <username>@lxplus.cern.ch:/etc/grid-security/vomsdir/cms /etc/grid-security/vomsdir/c
```

REST Interface installation and configuration

Installation

The following instructions are basically point "7. Software installation" of the HTTP group tutorial, where we updated the versions of the software being installed (the latest HG tags can be found in https://github.com/dmwm/deployment/releases).

Get the configuration and deploy the `frontend`, `crabserver` and `crabcache` services (the following command will ask you to type your Grid user certificate passphrase twice):

```
HGVER="HGXXXXX"
(cd /data; git clone git://github.com/dmwm/deployment.git cfg && cd cfg && git reset --hard $HGVE
(REPO="-r comp=comp" A=/data/cfg/admin;
 cd /data;
 $A/InstallDev -R comp@$HGVER -A slc7_amd64_gcc630 -s image -v $HGVER $REPO -p "admin/devtools fr
```

Note: `crabserver` is the CRAB REST Interface, `crabcache` is the CRAB User File Cache and `frontend` is the service receiving the http requests and redirecting them to the corresponding (backend) service (it runs an HTTP daemon).

Configuration

External (dynamic) configuration

Prepare the external configuration file for the REST Interface.

The external REST configuration file should be in JSON format. It may contain more than one configuration. Each configuration is a dictionary identified by an (arbitrary) key.

Configuration example for the production REST Interface in CMSWEB (maintained in https://gitlab.cern.ch/crab3/CRAB3ServerConfig/):

```
{
    "cmsweb-prod": {
        "delegate-dn": [
            "/DC=ch/DC=cern/OU=computers/CN=vocms(045|052|021|031).cern.ch"
```

```
        ],
        "backend-urls" : {
            "cacheSSL" : "https://cmsweb.cern.ch/crabcache",
            "baseURL" : "https://cmsweb.cern.ch/crabcache",
            "htcondorSchedds" : {
                "crab3test-2@vocms095.cern.ch" : {
                    "proxiedurl": "https://cmsweb.cern.ch/scheddmon/095"
                },
                "crab3test-3@vocms096.cern.ch" : {
                    "proxiedurl": "https://cmsweb.cern.ch/scheddmon/096"
                },
                "crab3-2@vocms0109.cern.ch" : {
                    "proxiedurl": "https://cmsweb.cern.ch/scheddmon/0109"
                },
                "crab3-4@vocms066.cern.ch" : {
                    "proxiedurl": "https://cmsweb.cern.ch/scheddmon/066"
                }
            },
            "htcondorPool" : "vocms097.cern.ch,vocms099.cern.ch",
            "asoConfig" : [
                {"couchURL" : "https://cmsweb.cern.ch/crabserver/prod" , "couchDBName" : "filetra
            ]
        },
        "compatible-version" : ["3.3.14", "3.3.15", "3.3.16"],
        "banned-out-destinations" : ["T1_*"]
    }
}
```

Configuration example for the pre-production REST Interface in CMSWEB-testbed (maintained in https://gitlab.cern.ch/crab3/CRAB3ServerConfig/):

```
{
    "cmsweb-preprod": {
        "delegate-dn": [
            "/DC=ch/DC=cern/OU=computers/CN=vocms(045|052|021|031).cern.ch|/DC=ch/DC=cern/OU=comp
        ],
        "backend-urls" : {
            "cacheSSL" : "https://cmsweb-testbed.cern.ch/crabcache",
            "baseURL" : "https://cmsweb-testbed.cern.ch/crabcache",
            "htcondorSchedds" : {
                "crab3-5@vocms059.cern.ch" : {
                    "proxiedurl" : "https://cmsweb.cern.ch/scheddmon/059"
                }
            },
            "htcondorPool" : "vocms097.cern.ch,vocms099.cern.ch",
            "asoConfig" : [
                {"couchURL" : "https://cmsweb.cern.ch/crabserver/prod" , "couchDBName" : "filetra
            ]
        },
        "compatible-version" : ["3.3.8", "3.3.9.rc2", "3.3.9", "3.3.10"],
        "banned-out-destinations" : ["T1_*"]
    }
}
```

Configuration example for a private REST Interface (maintained in https://gitlab.cern.ch/crab3/CRAB3ServerConfig/):

```
{
    "cmsweb-dev": {
        "delegate-dn": [
            "<Grid-host-certificate-DN-of-VM-where-private-TaskWorker-is-installed>|<Grid-host-ce
        ],
        "backend-urls" : {
            "cacheSSL" : "https://cmsweb-testbed.cern.ch/crabcache",
            "baseURL" : "https://cmsweb-testbed.cern.ch/crabcache",
```

```
            "htcondorSchedds" : {
                "crab3-5@vocms059.cern.ch" : {
                    "proxiedurl" : "https://cmsweb.cern.ch/scheddmon/059"
                }
            },
            "htcondorPool" : "vocms097.cern.ch,vocms099.cern.ch",
            "asoConfig" : [
                {"couchURL" : "https://cmsweb.cern.ch/crabserver/prod" , "couchDBName" : "filetra
            ]
        },
        "compatible-version" : ["3.3.[0-9a-zA-Z\.]+"],
        "banned-out-destinations" : ["T1_*"]
    }
}
```

Here is an explanation of the meaning of each of these parameters:

`delegate-dn`	List of DNs that are allowed to retrieve the user's proxy from myproxy server. CRAB services use Grid host certificates to contact myproxy server. So `delegate-dn` should include the Grid host certificate DNs of the hosts where the corresponding instances (production, pre-production or private) of the TaskWorker and AsyncStageOut are running ("corresponding" instances means the TaskWorker instance that will work on the tasks submitted to this REST, and the AsyncStageOut instance(s) that these tasks will use).
`cacheSSL`, `baseURL`	URL of the `crabcache` instance to be used by this REST. If you want to use your private `crabcache`, change these parameters to point to your machine.
`htcondorSchedds`	Dictionary of schedds on the pool. The keys of the dictionary are the schedd names as shown by `condor_status -schedd`. The values for the schedds are a dictionaries that contains the configuration of that schedd. For each schedd it is possible to specify a parameter called `proxiedurl` that indicates the URL used for the dashboard logfiles and for the `crab status --short` command
`htcondorPool`	As from Nov 2014, everything is moved to the global pool and this parameter must be `vocms097.cern.ch,vocms099.cern.ch`. crabserver will split by ','.
`asoConfig`	REST URL and DB name for the (default) DB instance to be used by the CRAB server for ASO-related businesses, "filetransfers" DB means Oracle (the user can overwrite this config in the task CRAB configuration).
`compatible-version`	List of CRAB client versions that are compatible with this REST Interface instance. Can use python regular expressions. (If the client is not compatible, a warning message will be printed on every crab command. To avoid the warning message in private tests, you may simply allow any version.)
`banned-out-destinations`	List of sites where stage out is forbidden (e.g. it is not allowed to stage out to any T1).

Note: Under any doubt about what value should you assign to a parameter, please contact the CRAB3 team.

You can put this configuration file on gist (https://gist.github.com/). This is what all CRAB3 developers do and is assumed you will do as well.

Internal (static) configuration

The internal REST configuration file is `/data/srv/current/config/crabserver/config.py`

The internal REST configuration contains a link to the external REST configuration file. So edit the internal REST configuration file and add the link to your external REST configuration file (specify also the key of the configuration that should be used).

```
sudo vi /data/srv/current/config/crabserver/config.py
```

Example for the external REST configuration file in https://git.cern.ch/web/CRAB3ServerConfig.git used by the production and pre-production REST Interfaces:

```
data.extconfigurl = 'http://git.cern.ch/pubweb/?p=CRAB3ServerConfig.git;a=blob_plain;f=cmsweb-res
data.mode = 'cmsweb-prod' # or 'cmsweb-preprod'
```

Example for an external REST configuration file corresponding to a private REST Interface:

```
data.extconfigurl = 'https://gist.githubusercontent.com/juztas/3e91e42dda800bb8c835/raw/67898398a
data.mode = 'cmsweb-dev' # or whatever is the key of your configuration
```

WARNING: The above link contains a revision number (the field between `raw` and the file name). This implies that every time the file in gist is changed, one needs to get the new link and change the URL in the static REST configuration file. To avoid this, one can use a link that points always to the latest version of the file on gist by removing the revision number:

```
data.extconfigurl = 'https://gist.githubusercontent.com/juztas/3e91e42dda800bb8c835/raw/gistfile1
data.mode = 'cmsweb-dev' # or whatever is the key of your configuration
```

Authentication

Authentication with CERN Oracle database

The REST authentication file for accessing the Oracle database is
`/data/srv/current/auth/crabserver/CRABServerAuth.py`

For private installations, modify the file to have the required information (don`t forget to change the oracle credentials (*********)).

```
sudo vi /data/srv/current/auth/crabserver/CRABServerAuth.py
```

```
import cx_Oracle as DB
import socket
fqdn = socket.getfqdn().lower()
dbconfig = {'dev': {'.title': 'Pre-production',
                    '.order': 1,
                    '*': {'clientid': 'cmsweb-dev@%s' % (fqdn),
                          'dsn': 'devdb11',
                          'liveness': 'select sysdate from dual',
                          'password': '***************',
                          'schema': '**********',
                          'timeout': 300,
                          'trace': True,
                          'type': DB,
                          'user': '*********'}
                   }
           }
```

Parameters explanation:

`title`, `order`	Any string will do.
`clientid`, `dsn`, `liveness`, `timeout`, `trace`, `type`	Must be like indicated.
`user`	Your CERN Oracle account username.
`password`	Your CERN Oracle account password.
`schema`	Put your CERN Oracle account username here as well.

Make sure the `CRABServerAuth.py` file is owned by `_sw _config`:

```
ls -l /data/srv/current/auth/crabserver/CRABServerAuth.py
```

```
-r--r-----. 1 _sw _config 622 Jun 15 09:30 /data/srv/current/auth/crabserver/CRABServerAuth.py
```

If not, do

```
sudo chown _sw:_config /data/srv/current/auth/crabserver/CRABServerAuth.py
```

Renewal

You will have to update the `CRABServerAuth.py` file every year with the new oracle password.

Authentication with frontend

ONLY for PRIVATE installations.

1) Check that your user DN is present in `/data/srv/state/frontend/etc/authmap.json`. If you want other users to be able to access your REST Interface, add their DNs to this file.

2) As part of the `frontend` installation, a cron job is created (which runs every 4 hours) that executes `/data/srv/current/config/frontend/mkauthmap -q -c sitedbread.db -o /data/srv/state/frontend/etc/authmap.json`. This cron job updates the `/data/srv/state/frontend/etc/authmap.json` file with the DNs from all users as obtained from querying the `sitedb` service. If you are not running the `sitedb` service in your VM, the cron job will not fail, but update the `/data/srv/state/frontend/etc/authmap.json` file with only your user DN. On the other hand, if you are running the `sitedb` service in your VM, the updated file will contain the DNs from all users. Thus, for private installations, we recommend to comment out this "automatic role setting" in the crontab:

```
crontab -e

#*/4 * * * * . /data/srv/current/apps/frontend/etc/profile.d/init.sh && PYTHONPATH=/data/srv/curr
```

Certificate for interactions with CMSWEB

Access to CMSWEB is restricted to CMS users and services by requesting authentication with certificates registered in VO CMS and SiteDB. The static REST configuration file has two parameters to point to a certificate and private key that the REST Interface should use for interactions with CMSWEB:

```
data.serverhostcert = "%s/auth/crabserver/dmwm-service-cert.pem" % __file__.rsplit('/', 3)[0]
data.serverhostkey = "%s/auth/crabserver/dmwm-service-key.pem" % __file__.rsplit('/', 3)[0]
```

where `__file__ = /data/srv/current/config/crabserver/config.py` and therefore `__file__.rsplit('/', 3)[0] = /data/srv/current/`.

Some of the CMS services use Grid service certificates for interactions with CMSWEB, but the majority, and in particular the production and pre-production CRAB services, use the operator's proxy. The reasons are both for convenience and security. For private installations you are the operator, so you should use your own user proxy:

```
voms-proxy-init --voms cms --valid 192:00
sudo cp /tmp/x509up_u$UID /data/srv/current/auth/crabserver/dmwm-service-cert.pem
sudo cp /tmp/x509up_u$UID /data/srv/current/auth/crabserver/dmwm-service-key.pem
```

The proxy is created for 8 days (192 hours), because this is the maximum allowed duration of the VO CMS extension. Thus, the proxy has to be renewed every 7 days (at least). You can do it manually (executing the last three commands) or you can set up an automatic renewal procedure like is being done in production and pre-production.

Make sure the certificate and private key are owned by `_sw _config` (or `_crabserver _crabserver` for newer HG tags):

```
ls -l /data/srv/current/auth/crabserver/dmwm-service-*

-r--r-----. 1 _sw _config 5233 Jun  1 19:15 /data/srv/current/auth/crabserver/dmwm-service-cert.p
-r--r-----. 1 _sw _config 5233 Jun  1 19:15 /data/srv/current/auth/crabserver/dmwm-service-key.pe
```

If not, do

```
sudo chown _sw:_config /data/srv/current/auth/crabserver/dmwm-service-{cert,key}.pem
```

Note: Grid host certificates can not be used as Grid service certificates if they are not registered in VO CMS and SiteBD.

Using your own CRABServer (and WMCore) repository

If you are a developer, most probably you want to use your own repositories.

1) Clone the repositories.

Fork the dmwm/CRABServer and dmwm/WMCore repositories on github to have them under your github username (you need to have a github account) and clone them.

Note: The instructions below suggest to put the cloned repositories into the `/data/user/` directory of your host. But if you will install the REST Interface and the TaskWorker on different hosts, then you should better use another place that is accessible from both hosts, e.g. AFS.

- CRABServer: https://github.com/dmwm/CRABServer --- Fork on github ---> https://github.com/<your-github-username>/CRABServer

```
cd /data/user/
git clone https://github.com/<your-github-username>/CRABServer
cd CRABServer
git remote -v

origin   https://github.com/<your-github-username>/CRABServer (fetch)
origin   https://github.com/<your-github-username>/CRABServer (push)

git remote add upstream https://github.com/dmwm/CRABServer
git remote -v

origin   https://github.com/<your-github-username>/CRABServer (fetch)
origin   https://github.com/<your-github-username>/CRABServer (push)
upstream   https://github.com/dmwm/CRABServer (fetch)
upstream   https://github.com/dmwm/CRABServer (push)
```

- WMCore: https://github.com/dmwm/WMCore --- Fork on github ---> https://github.com/<your-github-username>/WMCore

```
cd /data/user/
git clone https://github.com/<your-github-username>/WMCore
cd WMCore
git remote -v

origin   https://github.com/<your-github-username>/WMCore (fetch)
origin   https://github.com/<your-github-username>/WMCore (push)

git remote add upstream https://github.com/dmwm/WMCore
```

```
git remote -v

origin   https://github.com/<your-github-username>/WMCore (fetch)
origin   https://github.com/<your-github-username>/WMCore (push)
upstream   https://github.com/dmwm/WMCore (fetch)
upstream   https://github.com/dmwm/WMCore (push)
```

Each `crabserver` release uses a given version of WMCore as specified in https://github.com/cms-sw/cmsdist/blob/comp_gcc493/crabserver.spec Unless you will use an old tag of CRABServer, you should use the given WMCore tag:

```
git checkout <tag> # e.g. 1.0.12_crab_1
```

2) Configure `crabserver` to use your repositories.

In the `crabserver` init script, fix the setting of the `PYTHONPATH` to point to your cloned CRABServer and WMCore repositories.

Edit the file `/data/srv/current/sw.pre/*/cms/crabserver/*/etc/profile.d/init.sh`, comment out the lines that are changing the `PYTHONPATH` (should be the last two lines) and add a new line at the end of the script adding your repositories to the `PYTHONPATH` (here we assume the repositories are under `/data/user/`):

```
#[ ! -d /data/srv/HG1505c/sw.pre/slc7_amd64_gcc630/cms/crabserver/3.3.16.rc2/${PYTHON_LIB_SITE_PA
#[ ! -d /data/srv/HG1505c/sw.pre/slc7_amd64_gcc630/cms/crabserver/3.3.16.rc2/x${PYTHON_LIB_SITE_P
export PYTHONPATH=/data/user/CRABServer/src/python/:/data/user/WMCore/src/python/:$PYTHONPATH
```

Install/Scratch/Update the Oracle database tables

1) Create a configuration for connecting to the Oracle database, needed to install the database tables.

```
vi /data/dbconfig.py
```

Copy/paste the following content (put your `oracle_username` and `oracle_password`):

```
from WMCore.Configuration import Configuration
config = Configuration()
config.section_('CoreDatabase')
config.CoreDatabase.connectUrl = 'oracle://<oracle_username>:<oracle_password>@devdb11'
```

2) Set the environment.

```
source /data/srv/current/sw.pre/slc7_amd64_gcc630/cms/crabserver/3.3.16.rc2/etc/profile.d/init.sh
source /afs/cern.ch/project/oracle/script/setoraenv.sh -s prod
```

3) Install the required database tables (you have to clone WMCore from github so you have the `wmcore-db-init` command).

```
/data/user/WMCore/bin/wmcore-db-init --config /data/dbconfig.py --create --modules=Databases.Task
```

4) To scratch the database you will need to run the following commands:

```
export PYTHONPATH=/data/user/WMCore/src/python/:$PYTHONPATH #we need additional WMCore modules
/data/user/WMCore/bin/wmcore-db-init --config /data/dbconfig.py --destroy --modules=Databases.Tas
```

N.B.: the --destroy command will be available in WMCore once this pull is merged: https://github.com/dmwm/WMCore/pull/6055

5) To update the database when new code requires new columns in Task data base, the DB table needs to be modified by hand. The SQL commands to execute are added to updateOracle.sql by the developer who introduces the DB changes. Those command can be e.g. copied and pasted to the Oralcle SQL prompt on lxplus:

```
source /afs/cern.ch/project/oracle/script/setoraenv.sh -s prod
sqlplus <username>@devdb11
or
sqlplus <username>/<password>@devdb11
```

Example:

Show Hide

```
belforte@lxplus0067/ISTRUZ> source /afs/cern.ch/project/oracle/script/setoraenv.sh -s prod
belforte@lxplus0067/ISTRUZ> sqlplus crab3_belforte/c3s-1705@devdb11

SQL*Plus: Release 11.2.0.3.0 Production on Fri Jul 24 11:59:04 2015

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, Real Application Clusters, OLAP, Data Mining
and Real Application Testing options

SP2-0310: unable to open file "?/sqlplus/admin/glogin.sql"
SQL> select table_name from user_tables;

TABLE_NAME
------------------------------
TASKS
FILEMETADATA
FILETRANSFERSDB
JOBGROUPS

SQL>
SQL> alter table tasks add (tm_user_files CLOB DEFAULT '[]');
alter table tasks add (tm_user_files CLOB DEFAULT '[]')
                       *
ERROR at line 1:
ORA-01430: column being added already exists in table
[EDITOR NOTE: IT IS OK TO TRY TO ADD A  COLUMN WHICH IS THERE ALREADY]

SQL> alter table tasks add (tm_publish_groupname VARCHAR(1) DEFAULT 'F');

Table altered.

SQL> alter table tasks add constraint check_tm_publish_groupname check (tm_publish_groupname IN (

Table altered.

SQL> alter table tasks add (tm_nonvalid_input_dataset VARCHAR(1) DEFAULT 'T');
alter table tasks add constraint ck_tm_nonvalid_input_dataset check (tm_nonvalid_input_dataset IN

Table altered.

SQL>
Table altered.

SQL> quit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, Real Application Clusters, OLAP, Data Mining
```

```
and Real Application Testing options
belforte@lxplus0067/ISTRUZ>
```

Start/stop the REST Interface

Create a file named `crabserver.sh` with the following content:

```
numargs=${#}
if [ $numargs -eq 0 ]; then
  echo "Please specify the action (start, stop or status) to take on local CRAB server."
elif [ $numargs -eq 1 ]; then
  action=${1}
  cwd=$(pwd)
  echo "Sourcing environment from /data/srv"
  source /data/srv/current/sw.pre/slc7_amd64_gcc630/cms/crabserver/3.3.16.rc2/etc/profile.d/init.
  echo "Executing cd /data; /data/cfg/admin/InstallDev -d /data/srv -s $action; cd $cwd; stty san
  cd /data; /data/cfg/admin/InstallDev -d /data/srv -s $action; cd $cwd; stty sane
else
  echo "This script takes only 1 argument ($numargs arguments were given)."
fi
```

Now you can source the `crabserver.sh` script to perform one of the following actions on the `crabserver`, `crabcache` and `frontend` services:

- Start:

```
source crabserver.sh start
```

Check that you can access the frontend using the URL https://<host-name>/ You will be able to access the services you installed. In particular, https://<host-name>/crabserver/dev/workflow, etc.

- Get the status:

```
source crabserver.sh status
```

- Stop:

```
source crabserver.sh stop
```

Special settings of the Oracle production database

In the production database for security reason SELECT and UPDATE/INSERT are performed by different users with appropriate readonly and writeonly privileges, namely `cms_analysis_reqmgr_r` and `cms_analysis_reqmgr_w`. Every request that is done using a `GET` will use the reader account, every `POST` or `PUT` will use the writer account.

If you add a table you need to add the following privileges as well (not tested):

```
CREATE ROLE CMS_CRAB3_LOCAL_READER_ROLE;
CREATE ROLE CMS_CRAB3_LOCAL_WRITER_ROLE;
GRANT CMS_CRAB3_LOCAL_READER_ROLE TO CMS_ANALYSIS_REQMGR_R;
GRANT CMS_CRAB3_LOCAL_WRITER_ROLE CMS_ANALYSIS_REQMGR_W;
GRANT SELECT ON TABLE_NAME TO CMS_CRAB3_LOCAL_READER_ROLE;
GRANT INSERT, UPDATE, DELETE ON TABLE_NAME TO CMS_CRAB3_LOCAL_WRITER_ROLE;
```

You also need to create synonyms on the reader and witer accounts for the added table (TODO document).

Log files

Supposing the setup has been done under the `/data/srv` directory, the log files of the REST interface are located under `/data/srv/logs/crabserver/`.

The log file shows all HTTP that receives and shows the resources called, the returned status code, the calling IP and DN. Of course, also the date is associated for every log entry. As example:

```
[12/Jul/2013:15:14:15] mattia-dev01 188.184.21.35 "GET /crabserver/dev/workflowdb?getstatus=HOLDI
```

The Apache frontend logs can instead be found under `/data/srv/logs/frontend/`.

Pre-Installed CRABSERVER

A development version of crabserver is available thanks to Todor and Emilis. `crab-dev-rest.cern.ch` which is an alias for `vocms035.cern.ch`

This can be used for quick checks w/o need to install a personal one. Check with other people before changing things !

- to submit to it: `config.General.instance = 'vocms035.cern.ch'` (can't use alias here)
- to operate on it, `ssh` and then `sudo su crab3`
- repository: crab server code comes from `/user/data` which is cloned from Emilis' GIT
- start/stop/check server with : `/etc/init.d/crabserver start/stop/status`
- connected to dev TW: `crab-dev-tw01.cern.ch` which is an alias for `vocms058.cern.ch`

Tips and tricks

Client fails to contact server with No such instance error

In the server log you can usually find:

```
[11/Jul/2013:10:07:03]  SERVER REST ERROR WMCore.REST.Error.NoSuchInstance 30cacdcf7522dbb72b1310
[11/Jul/2013:10:07:03]    Traceback (most recent call last):
[11/Jul/2013:10:07:03]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:07:03]        return self._call(RESTArgs(list(args), kwargs))
[11/Jul/2013:10:07:03]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:07:03]        self._precall(param)
[11/Jul/2013:10:07:03]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:07:03]        raise NoSuchInstance()
[11/Jul/2013:10:07:03]    NoSuchInstance
[11/Jul/2013:10:07:03] c3p1 137.138.210.210 "GET /crabserver/dev/info?subresource=delegatedn HTTP
```

If the client is correctly performing calls to the server, this can be a symptom of misconfiguration in the database configuration file `/data/srv/current/auth/crabserver/CRABServerAuth.py`

Client fails to contact server with You are not allowed to access this resource error

In the server log you can usually find:

```
[11/Jul/2013:10:09:44]  SERVER HTTP ERROR cherrypy._cperror.HTTPError f0aa48dfd2f9d215c2214434494
[11/Jul/2013:10:09:44]    Traceback (most recent call last):
[11/Jul/2013:10:09:44]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:09:44]        return self._call(RESTArgs(list(args), kwargs))
```

```
[11/Jul/2013:10:09:44]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:09:44]        v(apiobj, request.method, api, param, safe)
[11/Jul/2013:10:09:44]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:09:44]        authz_login_valid()
[11/Jul/2013:10:09:44]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:09:44]        raise cherrypy.HTTPError(403, "You are not allowed to access this r
[11/Jul/2013:10:09:44]    HTTPError: (403, 'You are not allowed to access this resource.')
[11/Jul/2013:10:09:44] c3p1 137.138.210.210 "GET /crabserver/dev/info?subresource=delegatedn HTTP
```

This is usually a symptom of misconfiguration of a **private deployment**. In a private deployment it is needed to set up the users that are allowed to access the frontend in the `/data/srv/state/frontend/etc/authmap.json` file. If you do not disable the crontab entry as indicated in Authentication with frontend, then the `authmap.json` file will be rewritten with only your DN.

Note: This applies exclusively to private deployments and not to the CMSWEB environment.

Client fails to contact server with Impossible to retrieve proxy from myproxy.cern.ch for /DN/... error

In the server log you can usually find:

```
[11/Jul/2013:10:14:14]  SERVER REST ERROR WMCore.REST.Error.InvalidParameter 6ab2195b14aa1cc3d192
[11/Jul/2013:10:14:14]    Traceback (most recent call last):
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        return self._call(RESTArgs(list(args), kwargs))
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        obj = apiobj['call'](*safe.args, **safe.kwargs)
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        self._dberror(e, format_exc(), False)
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        return handler(*xargs, **xkwargs)
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        edmoutfiles=edmoutfiles, runs=runs, lumis=lumis, totalunits=totalun
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        return self.workflow.submit(*args, **kwargs)
[11/Jul/2013:10:14:14]      File "/data/hg1307b/sw.pre.spiga/slc5_amd64_gcc461/cms/crabserver/3.2
[11/Jul/2013:10:14:14]        raise invalidp
[11/Jul/2013:10:14:14]    InvalidParameter
[11/Jul/2013:10:14:14] c3p1 137.138.210.210 "PUT /crabserver/dev/workflow HTTP/1.1" 400 [data: 63
```

As said in the previous sections, the CRAB REST interface needs to access MyProxy, and in this case it is failing, because it cannot retrieve the user proxy from MyProxy. The source of this failure is not unique:

Please check the value of the `data.serverhostcert` and `data.serverhostkey` parameters: by default they point to dmwm-service-key.pem, which is what cmsweb expects but does not work for private installations. Please set these two parameters to:

```
data.serverhostcert = '/data/srv/current/auth/crabserver/hostcert.pem'
data.serverhostkey = '/data/srv/current/auth/crabserver/hostkey.pem'
```

Also, another possible solution for "Impossible to retrieve proxy" error is using your own proxy instead of the hostcert/hostkey. You can do that by modifying the external configuration and adding your user DN to the delegate-dn field so that it looks something like this:

```
delegate-dn": [
            "/DC=ch/DC=cern/OU=computers/CN=erupeikavm.cern.ch|/DC=ch/DC=cern/OU=Organic Units/OU
        ]
```

- The proxy of the user has not been delegated to MyProxy; there is an utility to help the debugging of such issues which can be used as following:

```
[15:27:04][mcinquil@mattia-dev01] ~> source /data/srv/current/apps/crabserver/etc/profile.
[15:28:03][mcinquil@mattia-dev01] ~> wget --no-check-certificate https://raw.github.com/dm
--2013-07-12 15:28:10--  https://raw.github.com/dmwm/CRABServer/master/bin/logon_myproxy_o
Resolving raw.github.com... 199.27.76.133
Connecting to raw.github.com|199.27.76.133|:443... connected.
WARNING: cannot verify raw.github.com's certificate, issued by `/C=US/O=DigiCert Inc/OU=ww
  Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 200 OK
Length: 1000 [text/plain]
Saving to: `logon_myproxy_openssl.py'

100%[=======================================================================================

2013-07-12 15:28:10 (19.1 MB/s) - `logon_myproxy_openssl.py' saved [1000/1000]

[15:28:10][mcinquil@mattia-dev01] ~> python logon_myproxy_openssl.py
Usage example:
python logon_openssl.py "/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=mcinquil/CN=660800/CN
```

What to do if your database is growing too much

This is the simple procedure and list of commands you need to execute to clean everything older than 1 month from the filemetadata table.

```
/afs/cern.ch/user/m/mmascher/scripts/database > sqlplus
crab_gwms@devdb11

SQL*Plus: Release 11.2.0.3.0 Production on Fri May 9 01:34:37 2014

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SP2-0310: unable to open file "?/sqlplus/admin/glogin.sql"

SQL> select * from user_ts_quotas;

TABLESPACE_NAME             BYTES  MAX_BYTES     BLOCKS MAX_BLOCKS DRO
------------------------------ ---------- ---------- ---------- ---------- ---
DATA01                  51642368   52428800       6304       6400 NO

SQL> delete from filemetadata where fmd_creation_time < ADD_MONTHS
(SYSDATE, -1);

30034 rows deleted.

SQL> PURGE RECYCLEBIN;

Recyclebin purged.

SQL> alter table filemetadata enable row movement;

Table altered.

SQL> alter table filemetadata shrink space;

Table altered.

SQL> select * from user_ts_quotas;

TABLESPACE_NAME             BYTES  MAX_BYTES     BLOCKS MAX_BLOCKS DRO
```

```
------------------------------ ---------- ---------- ---------- ---------- ---
DATA01                         30539776   52428800        3728       6400 NO
```

This topic: CMSPublic > CMSCrabRESTInterfaceSlc7
Topic revision: r6 - 2018-03-16 - TitasKulikauskas