Automating the installation of ACS 2020-DEC on Debian 10 with Ansible

C. Ortega, F. Beltrán, P. de Vicente, P. Collado

Informe Técnico IT-CDT 2021-8

Revision history

Version	Date	Author	Updates
1.0	26-05-2021	C. Ortega, F. Beltrán, P. de Vicente & P. Collado	First version

Contents

1 Introduction

The ALMA Common Software (ACS hereafter) is the open source software infrastructure used at the ALMA Interferometer and at some astronomical telescopes. ACS, developed by the European Southern Observatory (ESO) with the project community's contribution, consists of a set of common patterns and a series of components that implement those patterns.

Even though ACS is officially supported only in RHEL 7.6 and CentOS 7.6, the installation can be a tedious process and requires a non-negligible amount of time because it requires compiling and installing numerous tools. Since Yebes Observatory does not use any of the previous Linux distributions, the installation of ACS requires tuning some scripts and preinstalling some (Debian) packages, so simplifying and making this procedure easy and reproducible in several Debian hosts is a necessity. To achieve this goal, using Ansible is the appropriate option. This report briefly describes the main characteristics of Ansible and then exposes the implementation of this tool together with ACS.

2 Non-automatic installation of ACS 2020-DEC on Debian 10

In this section it is described briefly the non-automatic installation of ACS 2020-DEC on a host running Linux Debian 10.

At the beginning, it is essential to download the required packages as `superuser`, maintaining these privileges until it is indicated. But, previously, it is convenient to check that the `/etc/apt/sources.list` file contains the following repositories:

```
deb http://deb.debian.org/debian/ buster main
deb-src http://deb.debian.org/debian/ buster main
deb http://security.debian.org/debian-security buster/updates main
deb-src http://security.debian.org/debian-security buster/updates main
deb http://deb.debian.org/debian/ buster-updates main
deb-src http://deb.debian.org/debian/ buster-updates main
```

After adding a repository, it is obligatory to reload the package manager by:

```
apt update
```

Now it is time to install the corresponding packages with the `apt install` command. The complete list of all of them is:

```
ksh gcc g++ gfortran python-dev libx11-dev libxml2-dev libxslt1-dev zlib1g-dev doxygen valgrind
procmail openjdk-11-jdk ant maven python-virtualenv libffi-dev perl bzip2 libxml2-dev libxslt1-dev
bison flex autoconf unzip dos2unix tcl-dev tk-dev lsb-base openssl sqlite3 expat build-essential
libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev
libncursesw5-dev xz-utils libffi-dev liblzma-dev python-openssl python3-openssl git git-lfs
python-bz2file python3-bz2file libedit-dev libzip-dev libldap2-dev libfreetype6-dev time omniidl
rsync vim python-pip python3-pip python-sqlite libsasl2-dev 2to3 libhdf5-de xterm subversión
libcfitsio-dev libjfreechart-java linux-source
```

Then, in the `root` directory, it is created the directory where the ACS installation will be placed and use permissions are given to the normal user of the device to be able to work with it.

```
mkdir /alma
chown -R ortega.ortega /alma
```

A symbolic link should be made in the `/usr/bin` directory with the `gtar` name and pointing to `tar` to avoid any reference to `gtar`.

```
ln -s tar gtar
```

After the previous steps, no more superuser privileges are necessary and the coming actions are executed as a normal user. Thus, in the `HOME` of the mentioned user, a repository is cloned and a symbolic link to the downloaded files is generated, proceeding this way:

```
cd $HOME
git clone http://git.oan.es/acs-control-systems/acs-2020dec.git ACS-2020DEC
ln -s ACS-2020DEC ACS
```

To avoid errors during the compilation process, both documents `$HOME/ACS-2020DEC/ExtProd/INSTALL/standardUtilities` and `$HOME/ACS-2020DEC/LGPL/acsBUILD/config/.acs/.bash_profile.acs` should be edited and all occurrences with `"debian-"` should be replaced by `"debian_"`.

Also, in the `$HOME/ACS-2020DEC/ExtProd/PRODUCTS` directory, inside the `acs-py27.req` and `acs-py37.req` files, the `ephem` version should be replaced by `3.7.7.0` and the `astropy` package appended, as well.

Next, it is fundamental to initialize the needed environment variables like this:

```
cd $HOME
mkdir $HOME/.acs
cp $HOME/ACS/LGPL/acsBUILD/config/.acs/.bash_profile.acs $HOME/.acs/
source $HOME/.acs/.bash_profile.acs -r
```

Furthermore, it is essential to include in the `$HOME/.bashrc` document the following lines:

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
source $HOME/.acs/.bash_profile.acs -r
export CLASSPATH=/usr/share/java/jfreechart.jar:"${CLASSPATH}"
```

To guarantee that the previous tasks are applied properly, the terminal should be restarted.

Later, the External Products are installed by invoking the commands:

```
cd $HOME/ACS/ExtProd/INSTALL
make all
```

The output should look like this:

```
WARNING: Do not close this terminal: some build might fail!
WARNING: DISPLAY not set. Some build/tests might fail!
Create ACS-2020DEC
2021-04-15T06:31:25 buildTclTk [2m39.680s]                [  OK  ]
2021-04-15T06:34:04 buildTAO [23m37.932s]                 [  OK  ]
2021-04-15T06:57:42 buildMaven [0m0.084s]                 [  OK  ]
2021-04-15T06:57:42 buildJavaDependencies [0m49.469s]     [  OK  ]
2021-04-15T06:58:32 buildAnt [0m19.545s]                  [  OK  ]
2021-04-15T06:58:51 buildJacorb ==== Building JacORB from: /home/.. into: /alma/ACS-2020DEC
==== Copying sources
==== Cumulative Patch
==== JacORB Version Patch
========= Done with unpacking and patching, can compile now... ==========
==== Building jars
==== Build Notification service IDL
==== Copy extra IDLs CosProperty(Service), DsLogAdmin and AVStreams into JacORB directory tree
==== Build with extra IDLs
======== SUCCESS!!!==========
[1m25.681s]                                               [  OK  ]
2021-04-15T07:00:17 buildPython [2m22.619s]               [  OK  ]
2021-04-15T07:02:40 buildPyModules [1m48.034s]            [  OK  ]
2021-04-15T07:04:28 buildOmniORB [7m16.860s]              [  OK  ]
2021-04-15T07:11:45 buildEclipse [0m1.690s]               [  OK  ]
2021-04-15T07:11:46 buildSwig [0m42.638s]                 [  OK  ]
2021-04-15T07:12:29 buildBoost [7m56.174s]                [  OK  ]
WARNING: Now log out and login again to make sure that
         the environment is re-evaluated!

__oOo__
2021-04-15T07:20:25 buildTools script has finished
. . . 'all' done
```

To check that the process has run flawlessly it is convenient to look at the generated `.log` files looking for the string `ERROR`.

Again, the terminal should be restarted to validate several environment variables created.

The ACS compilation should be done within the ACS directory and requires exporting the `INTROOT` environment variable previously:

```
cd $HOME/ACS
export INTROOT=$ACSROOT
make build
```

The screen output will look like this:

```
Evaluating current ACS TAG from http://git.oan.es/acs-control-systems/acs-2020dec.git
REPO tag is: acs/2020DEC
############ Clean Build Log File: build.log #################
############ Check directory tree for modules  #################
############ Prepare installation areas      #################
############ (Re-)build ACS Software         #################
############ LGPL/Kit/doc SRC
############ LGPL/Kit/acs SRC
############ LGPL/Kit/acstempl SRC
############ LGPL/Kit/acsutilpy SRC
############ LGPL/Tools MAIN
  ############ (Re-)build Tools Software         #################
  ############ tat MAIN
  ############ expat WS
  ############ loki WS
  ############ extjars MAIN
```

```
  ############ antlr MAIN
  ############ hibernate MAIN
  ############ extpy MAIN
  ############ cppunit MAIN
  ############ getopt MAIN
  ############ astyle MAIN
  ############ xercesc MAIN
  ############ xercesj MAIN
  ############ castor MAIN
  ############ xsddoc MAIN
  ############ extidl WS
  ############ vtd-xml MAIN
  ############ oAW MAIN
  ############ shunit2 MAIN
  ############ scxml_apache MAIN
  ############ DONE (Re-)build Tools Software    #################
############ LGPL/CommonSoftware/jacsutil SRC
############ LGPL/CommonSoftware/xmljbind SRC
############ LGPL/CommonSoftware/xmlpybind SRC
############ LGPL/CommonSoftware/acserridl WS
############ LGPL/CommonSoftware/acsidlcommon WS
############ LGPL/CommonSoftware/acsutil WS
############ LGPL/CommonSoftware/acsstartup SRC
############ LGPL/CommonSoftware/loggingidl WS
############ LGPL/CommonSoftware/logging WS
############ LGPL/CommonSoftware/acserr WS
############ LGPL/CommonSoftware/acserrTypes WS
############ LGPL/CommonSoftware/acsQoS WS
############ LGPL/CommonSoftware/acsthread WS
############ LGPL/CommonSoftware/acscomponentidl WS
############ LGPL/CommonSoftware/cdbidl WS
############ LGPL/CommonSoftware/maciidl WS
############ LGPL/CommonSoftware/baciidl WS
############ LGPL/CommonSoftware/acsncidl WS
############ LGPL/CommonSoftware/repeatGuard WS
############ LGPL/CommonSoftware/acsjlog SRC
############ LGPL/CommonSoftware/loggingts WS
############ LGPL/CommonSoftware/loggingtsTypes WS
############ LGPL/CommonSoftware/jacsutil2 SRC
############ LGPL/CommonSoftware/cdb WS
############ LGPL/CommonSoftware/cdbChecker SRC
############ LGPL/CommonSoftware/codegen SRC
############ LGPL/CommonSoftware/cdb_rdb SRC
############ LGPL/CommonSoftware/acsalarmidl WS
############ LGPL/CommonSoftware/acsalarm SRC
############ LGPL/CommonSoftware/acsContainerServices WS
############ LGPL/CommonSoftware/acscomponent WS
############ LGPL/CommonSoftware/recovery WS
############ LGPL/CommonSoftware/basenc WS
############ LGPL/CommonSoftware/archiveevents WS
############ LGPL/CommonSoftware/parameter SRC
############ LGPL/CommonSoftware/baci WS
############ LGPL/CommonSoftware/enumprop WS
############ LGPL/CommonSoftware/acscallbacks SRC
############ LGPL/CommonSoftware/acsdaemonidl WS
############ LGPL/CommonSoftware/jacsalarm SRC
############ LGPL/CommonSoftware/jmanager SRC
############ LGPL/CommonSoftware/maci WS
############ LGPL/CommonSoftware/task SRC
############ LGPL/CommonSoftware/acstime WS
############ LGPL/CommonSoftware/acsnc WS
############ LGPL/CommonSoftware/acsdaemon WS
############ LGPL/CommonSoftware/acslog WS
############ LGPL/CommonSoftware/acstestcompcpp SRC
############ LGPL/CommonSoftware/acsexmpl WS
############ LGPL/CommonSoftware/jlogEngine SRC
############ LGPL/CommonSoftware/acspycommon SRC
```

```
############ LGPL/CommonSoftware/acsalarmpy SRC
############ LGPL/CommonSoftware/acspy SRC
############ LGPL/CommonSoftware/comphelpgen SRC
############ LGPL/CommonSoftware/XmlIdl SRC
############ LGPL/CommonSoftware/define WS
############ LGPL/CommonSoftware/acstestentities SRC
############ LGPL/CommonSoftware/jcont SRC
############ LGPL/CommonSoftware/jcontnc SRC
############ LGPL/CommonSoftware/nsStatisticsService SRC
############ LGPL/CommonSoftware/jacsalarmtest SRC
############ LGPL/CommonSoftware/jcontexmpl SRC
############ LGPL/CommonSoftware/jbaci SRC
############ LGPL/CommonSoftware/monitoring MAIN
  ############ (Re-)build monitoring Software         #################
  ############ monicd WS
  ############ moncollect WS
  ############ monblobber MAIN
  ############ moncontroller MAIN
  ############ DONE (Re-)build monitoring Software    #################
############ LGPL/CommonSoftware/acssamp WS
############ LGPL/CommonSoftware/mastercomp SRC
############ LGPL/CommonSoftware/acspyexmpl SRC
############ LGPL/CommonSoftware/nctest WS
############ LGPL/CommonSoftware/acscommandcenter SRC
############ LGPL/CommonSoftware/acssim SRC
############ LGPL/CommonSoftware/bulkDataNT SRC
### ==> FAILED all !
### ==> FAILED install !
############ LGPL/CommonSoftware/bulkData SRC
############ LGPL/CommonSoftware/containerTests MAIN
############ (Re-)build(SUBSYSTEM) Software         #################
############ contLogTest MAIN
############ contNcTest MAIN
############ corbaRefPersistenceTest MAIN
############ contHandleTest MAIN
############ DONE (Re-)build containerTest Software    #################
############ LGPL/CommonSoftware/acscourse WS
############ LGPL/CommonSoftware/ACSLaser MAIN
  ############ (Re-)build Laser Software         #################
  ############ alarmCommon MAIN
  ############ laser-extlib MAIN
  ############ acs-jms MAIN
  ############ cmw-mom MAIN
  ############ laser-util MAIN
  ############ laser-source MAIN
  ############ laser-source-cpp MAIN
  ############ laser-source-python MAIN
  ############ gp-openide MAIN
  ############ gp MAIN
  ############ laser-core MAIN
  ############ alarmHibernate MAIN
  ############ laser-client MAIN
  ############ laser-definition MAIN
  ############ laser-console MAIN
  ############ alarm-clients MAIN
  ############ demo MAIN
  ############ managerTest MAIN
  ############ baciPropsTest MAIN
  ############ alarmTests MAIN
  ############ containerTest MAIN
  ############ DONE (Re-)build Laser Software    #################
############ LGPL/CommonSoftware/acsGUIs MAIN
  ############ (Re-)build ACS GUIs         #################
  ############ acsGUIutil MAIN
  ############ acsEclipseUtils MAIN
  ############ acssampGUI MAIN
  ############ cdbBrowser MAIN
```

```
  ############ errorBrowser MAIN
  ############ eventGUI MAIN
  ############ jlog MAIN
  ############ logLevelGUI MAIN
  ############ logTools MAIN
  ############ objexp MAIN
  ############ alarmSourcePanel MAIN
  ############ AlarmSystemProfiler MAIN
  ############ alarmPanel MAIN
  ############ DONE (Re-)build ACS GUIs    #################
############ LGPL/CommonSoftware/acsExtras MAIN
  ############ (Re-)build ACS EXTRAS         #################
  ############ acsXmlFileStore MAIN
  ############ DONE (Re-)build ACS EXTRAS    #################
############ Benchmark/util SRC
############ Benchmark/analyzer SRC
############ LGPL/acsBUILD SRC
############ FAILED (Re-)build ACS Software    #################
make: *** [Makefile:246: update] Error 1
```

A second time, in spite of the output seen, it is necessary to examine the `.log` files, searching in this case for coincidences with `FAILED`, only finding three related with `bulkDataNT`, all of them without significance and relevance and due to a paid library (`com.rti.dds`) taken into account in ACS and not available in the Yebes astronomical observatory.

If the earlier actions run faultlessly, the installation of ACS has been correct.

All the local software related to the 40m radio telescopes will be located in a different `introot` directory. This directory requires a special structure that can be obtained running:

```
cd
getTemplate
```

A graphical interface will pop up and will allow defining `directoryStructure`, `createINTROOTarea` and `introot` pressing three times the `INTRO` will set the default locations.

The code for the 40m RT radiotelescope is currently managed using Subversion as a version control system for software. To download the latest version of the code it is executed:

```
svn co http://hercules.oan.es/svn/trunk/aries21_2015.4
```

Previous to the compilation of the local software, to avoid compilation errors with the code, it must be added to the `$HOME/.bashrc` document before the lines inserted previously the following ones:

```
export INTROOT=$HOME/introot
export ACS_CDB=$HOME/aries21_2015.4/OANCDB
```

To force the usage of the system libraries included in a Debian standard installation the file `dist-packages.pth` in the directories `/alma/ACS-2020DEC/pyenv/versions/2.7.16/lib/python2.7/site-packages` and `/alma/ACS-2020DEC/pyenv/versions/3.6.9/lib/python3.6/site-packages` should contain the following lines, as appropriate, which allow the correct execution of ACS Python:

```
/usr/lib/python2.7
/usr/lib/python2.7/plat-x86_64-linux-gnu
/usr/lib/python2.7/lib-tk
/usr/lib/python2.7/lib-old
/usr/lib/python2.7/lib-dynload
/usr/local/lib/python2.7/dist-packages
/usr/lib/python2.7/dist-packages

/home/Ortega
/usr/lib/python37.zip
/usr/lib/python3.7
/usr/lib/python3.7/lib-dynload
/usr/local/lib/python3.7/dist-packages
/usr/lib/python3/dist-packages
```

Finally, it is necessary to install the GPIB Linux module in order to prevent mistakes in the control system during the compilation of some of the components. For this purpose, first, the installation package must be downloaded with:

```
wget https://sourceforge.net/projects/linux-gpib/files/latest/download -O linux-gpib.tar.gz
```

Then, the downloaded file and, at the same time, the documents that it contains, all of them should be decompressed by:

```
tar -xzvf file_name
```

In the directory with the `kernel` word in its name, it must be executed the next commands, the second one with `root` privileges:

```
make
make install
```

To conclude, in the directory with the `user` word in its name, it should be invoked the following commands, the last of them with `root` privileges:

```
./configure --sysconfdir=/etc
make
make install
```

3 What is Ansible?

Ansible is a very powerful IT automation tool that allows the configuration of systems, deployment of software and also cloud provisioning or continuous deployment. It stands out because of it simplicity and ease of use, being able to manage several devices in parallel, without loosing safety.

Ansible uses a simple language, YAML, that is very similar to English plain text, so it is easy for humans to read, write and understand. Ansible does not require any server, agent, daemon or database and works by default over the security strong protocol SSH.

Ansible needs a control node that could be a machine running any operating system, except Windows, with the mentioned tool and Python both installed. Nevertheless, remote systems do

not demand any program or application.

The code to execute in the administrator is generally written in documents called `playbooks`, but it could also be part of small executables named `ad-hoc commands`. Managed hosts, meanwhile, are listed in a file known as `inventory` or `hostfile`, too.

Ansible will execute the commands remotely in different devices using SSH. To ease this usage, the public SSH key of the control node should be included in the `authorized_keys` file in each of the accounts on such remote devices.

3.1 YAML (Yet Another Markup Language)

YAML is the language used to write Ansible files and it has positive characteristics, as well as, disadvantages:

- Documents in YAML can optionally begin with three dashes `("---")` and end with three points `("...")`.
- In the body of the file there may be something similar to a list. Each of its elements is a key/value pair. Between the key and the value there must be a colon followed by a space `(":  ")`.
- As expected, more elaborate data structures are possible, such as lists of dictionaries, dictionaries whose values are lists or a mix of both.
- To write multiple lines it is necessary to use a pipe `("|")` immediately after the colon and the space `(":  ")`.
- All items should be written in a new line with the same indentation level and, in lists, starting with a dash and a space `("- ")`.

So, in conclusion, YAML is very sensitive to indentation.

```
---
# A list
- item_1
- item_2
- item_3
- item_4

# A list of dictionaries
- dic_1:
    - item_1: value_1
    - item_2: value_2
    - item_3: value_3

- dic 2:
    - item_1: value_1
    - item_2: value_2
    - item_3: value_3

# Multiple lines
lines: |
```

```
    line_1
    line_2
    line_3
...
```

3.2 Inventory

The inventory is a simple text document that contains the list or group of lists of managed nodes. It is located at `/etc/ansible/hosts` and it can hold IP addresses or domain names. Also, more specific things are viable, such as creating aliases or setting variables. Hosts can belong to several groups, not only one. The related group name must be written between square brackets (`"[ ]"`).

In the playbook, targeted devices can be indicated with their group name in the inventory file. A SSH connection will be done from the administrator to every remote host in the list.

```
# A inventory
[group_name]
192.0.123.3
www.example.com
```

3.3 Playbooks

Playbooks are the main documents of Ansible. They are written using the YAML language and they look like a to-do list that contains single actions, called tasks, which may be organized in collections of actions, called plays. Each task invokes an Ansible module, a piece of code that runs in the managed nodes specified job and, which is removed automatically once it is executed. All of them run in order from top to bottom.

Specific parameters can be established at the playbook, play or task level to configure the proper behaviour of the whole environment. For instance, it is possible to indicate the group of devices to work with (`hosts` attribute), the user of the controlled hosts for the SSH connections (`remote_user` attribute), or apply the privilege escalation to do tasks as `sudo` or `su` method (`become` attribute and, optionally, `become_method` attribute).

To run those playbooks it is necessary to invoke the `ansible-playbook` command together with the playbook name, a file that could be any one ended with the `.yml` extension. Also, if the remote machines are more than five (the default value), it is recommended to mention the number of them with the command option `-f` to achieve a greater efficiency and, if a password is needed while granting privileges, it is required to add the `--ask-become-pass` or `-K` option.

When the execution is completed, a summary of the actions done and their results can be seen in the Ansible output. Errors can be omitted by means of the `ignore_errors` attribute of a task.

Playbooks also allow the usage of variables. Variables are identified with a particular word that is used to refer to them. Variables are included in a list following the rules named before and placed in the same file or in a separate document that can be included with the attributes `include_vars` or `vars_files`. To refer to these variables it is essential to utilise the structure `"{{ variable_name }}"` and, if it is convenient, enclose it between quotation marks, as well.

Variables can be used in loops and can be created during the execution of the playbook running to analyze a parameter in particular, such as the output of an action specifically. This is made by means of the `register` task attribute together with the `debug` task, in whose `msg` attribute the related variable is indicated.

For a better analysis of the whole playbook and to understand the functionality of each play or task, they allow to add a small description in their `name` attribute.

```
---
# A play
- name: a example of play
  hosts: name_group_hosts

  # A list of tasks
  tasks:
  # A task
  - name: print a message
    debug:
      msg: Hello world

  # Another task
  - name: copy a file from the source path to the destination path
    copy:
      src: source_path/file_name
      dest: destination_path/file_name

# Another play
- name: desc_play_2
  att_1: value_1
  att_2: value_2
  att_3: value_3

  # A list of tasks
  tasks:
  # A task
  - name: desc_task_1
    module_name:
      att_1: value_1
      att_2: value_2

  # Another task
  - name: desc_task_2
    module_name:
      att_1: value_1
      att_2: value_2
...
```

3.4 Roles

Roles are powerful mechanisms that allow breaking a complex or large playbook in small pieces of code to simplify its structure. This diversification permits to separate actions with

completely different characteristics for a better understanding, isolating each one of them and, also, if necessary, reusing one or several of these parts.

Roles can not be executed by themselves, they must be invoked from inside a playbook. For this purpose, the attribute `roles` is needed followed by the list of the existing ones.

Every single role should have a directory with its identical name in the `roles` directory that is located in the same place as the related playbook. Inside this directory with the name of the role must be, at least, another directory called `tasks` that, as its name suggests, contains the expected actions to be done. These jobs should be put in a file named `main.yml`, by agreement, following the instructions mentioned before in the previous subsection. Other directories can be created at the same level as the `tasks` directory, for instance, a `vars` directory for variables or a `templates` directory for templates. In those last directories there may be the corresponding document or documents, as appropriate.

Also, it is possible to establish variables in the playbook that will be used in the roles. In order to reach this objective, it is essential to set those variables before the `roles` attribute.

This directory structure is very important to keep in mind to achieve the properly running.

```
# Directory structure
playbook_name.yml
roles/
     role1_name/
               tasks/
                    main.yml
               templates/
                        templates_file_name.xxx
               vars/
                   vars_file_name.yml
     role2_name/
               tasks/
                    main.yml
               templates/
                        templates_file_name.xxx
               vars/
                   vars_file_name.yml
```

```
---
# A play
- name: desc_play_1
  att_1: value_1
  att_2: value_2
  att_3: value_3
  vars:
    var_1: value_1
  roles:
    - role1_name
    - role2_name
...
```

All the previous characteristics mentioned are the basic ones and only a small portion of Ansible, much more functionalities are possible with it.

4 Ansible case of study with ACS 2020-DEC on Debian 10

As seen, Ansible provides many opportunities that should be kept in mind. According to this, the installation of ACS is a good chance to put Ansible into practice and simplify this process to a great extent.

First of all, before going in depth in the meaning of each and every task done, it is helpful to display the current directory structure:

```
acs.yml
roles/
     root/
               tasks/
                    main.yml
               vars/
                   vars_file.yml
     user/
               tasks/
                    main.yml
               templates/
                         scriptmakeall.j2
                         scriptmakebuild.j2
```

Also, the inventory file (`/etc/ansible/hosts` document) includes the next content, with the `group_name` inserted later in the `hosts` attribute of the playbook:

```
[acs]
172.16.10.1
```

The playbook follows the approach of separating and isolating actions with different properties and peculiarities. In this way, as their names well indicate, on one hand, some tasks are executed as a privileged user while, furthermore, other jobs are run as a normal user without any privilege. In both parts it is defined the group of hosts to work with, the user in the managed devices for the SSH connection, a variable identical to the `remote_user` to utilise it in the roles of the playbook and, as expected, the existing roles, too.

```
---
- name: root role
  hosts: acs
  remote_user: almamgr
  become: yes
  become_method: su
  vars:
    user: almamgr
  roles:
    - root

- name: user role
  hosts: acs
  remote_user: almamgr
  vars:
    user: almamgr
  roles:
    - user
...
```

4.1 `Root` role

In the first section strictly speaking (`roles/root/tasks/main.yml` document), an important thing to address is the download of the required packages for the subsequent installation of the External Products. But, before this, it is necessary to confirm the presence (`state` attribute) of the needed repositories in the available list of them for the successful download of the mentioned packages. The lists of those repositories and packages, the same as the cited in the non-automatic procedure, are placed in an individual file that must be imported.

```
- name: creating the /alma directory
- name: adding variables
  include_vars: vars_file.yml

- name: adding the needed repositories
  apt_repository:
    repo: "{{ item }}"
    state: present
    update_cache: yes
  loop: "{{ repos }}"

- name: installing the needed packages
  apt:
    name: "{{ packages }}"
    state: present
```

Then, the directory where the installation of ACS will be located is created.

```
- name: creating the /alma directory
  file:
    path: /alma
    state: directory
    owner: "{{ user }}"
    group: "{{ user }}"
    recurse: yes
```

Finally in this part, a symbolic link called `gtar` is made to point to the `tar` tool because the first one is not feasible and, also, to allow for a better interoperability with the last of them.

```
- name: creating the gtar symbolic link
  file:
    src: /bin/tar
    dest: /bin/gtar
    state: link
```

4.2 `User` role

In this other subsection (`roles/user/tasks/main.yml` document), all the actions executed do not demand to the user any privilege escalation for the right operation of all of them, except a minimal crucial case at the end.

At the beginning, the installation files of the corresponding ACS version are downloaded in a directory (`dest` attribute) by cloning a specific branch (`version` attribute) of a repository only with its last commit (`depth` attribute) to get the most recent version of these documents.

```
- name: cloning the ACS repository
  git:
    repo: 'http://git.oan.es/acs-control-systems/acs-2020dec.git'
    dest: /home/{{ user }}/ACS-2020DEC
    force: yes
    depth: 1
```

The next step consists in redirecting a symbolic link to a new position that is the place named in the immediately previous task. For this purpose, Ansible requires to delete (`state` attribute) the existing symbolic link, so creating another one is essential.

```
- name: removing the ACS symbolic link
  file:
    path: /home/{{ user }}/ACS
    state: absent

- name: creating the ACS symbolic link
  file:
    src: /home/{{ user }}/ACS-2020DEC
    dest: /home/{{ user }}/ACS
    state: link
```

Then, it is fundamental to correct several significant mistakes by means of the `replace` module putting the string to modify in its `regexp` attribute and to add some packages as appropriate with the `lineinfile` module, as well, all of that to succeed in the following things whilst avoiding incompatibilities.

```
- name: correcting mistakes
  replace:
    path: /home/{{ user }}/ACS-2020DEC/ExtProd/INSTALL/standardUtilities
    regexp: 'debian-'
    replace: 'debian_'

- name: correcting mistakes
  replace:
    path: /home/{{ user }}/ACS-2020DEC/LGPL/acsBUILD/config/.acs/.bash_profile.acs
    regexp: 'debian-'
    replace: 'debian_'

- name: correcting mistakes
  replace:
    path: /home/{{ user }}/ACS-2020DEC/ExtProd/PRODUCTS/acs-py27.req
    regexp: 'ephem==3.7.6.0'
    replace: 'ephem==3.7.7.0'

- name: adding package
  lineinfile:
    path: /home/{{ user }}/ACS-2020DEC/ExtProd/PRODUCTS/acs-py27.req
    line: astropy

- name: correcting mistakes
  replace:
    path: /home/{{ user }}/ACS-2020DEC/ExtProd/PRODUCTS/acs-py37.req
    regexp: 'ephem==3.7.6.0'
    replace: 'ephem==3.7.7.0'

- name: adding package
  lineinfile:
    path: /home/{{ user }}/ACS-2020DEC/ExtProd/PRODUCTS/acs-py37.req
    line: astropy
```

The coming three jobs are related to the initialization of environment variables that will be indispensable in the future. Thus, in the second one a file from the controlled machine itself (`remote_src` attribute) is copied and in the third of them a couple of lines are included in the `path` attribute document.

```
- name: creating the .acs directory
  file:
    path: /home/{{ user }}/.acs
    state: directory

- name: copying .bash_profile.acs
  copy:
    src: /home/{{ user }}/ACS/LGPL/acsBUILD/config/.acs/.bash_profile.acs
    dest: /home/{{ user }}/.acs
    remote_src: yes

- name: initializing environment variables
  blockinfile:
    path: /home/{{ user }}/.bashrc
    block: |
      export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
      source $HOME/.acs/.bash_profile.acs -r
```

After all of the above, a template or script (`scriptmakeall.j2`) is used to elude several difficulties associated with the necessary environment variables. Therefore, at the start, the cited script is moved to the managed systems by the `template` module to be run via the `cmd` attribute of the `shell` module and lastly, in order to see its output, the `debug` module utilises the variable registered just before. The main function of this script is to invoke the `make all` command for the installation of the External Products.

The content of the mentioned script is:

```
#!/bin/bash
export ALMASW_ROOTDIR=/alma
export ALMASW_RELEASE=ACS-2020DEC
export CYGWIN_VER=CYGWIN_NT-5.1
export ALMASW_INSTDIR=/alma/ACS-2020DEC
export ACSDATA=/alma/ACS-2020DEC/acsdata
export ACE_ROOT=/alma/ACS-2020DEC/TAO/ACE_wrappers/build/linux
export ACE_ROOT_DIR=/alma/ACS-2020DEC/TAO/ACE_wrappers/build
export M2_HOME=/alma/ACS-2020DEC/maven
export ANT_HOME=/alma/ACS-2020DEC/ant
export JACORB_HOME=/alma/ACS-2020DEC/JacORB
export PYENV_ROOT=/alma/ACS-2020DEC/pyenv
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
source /home/{{user}}/.acs/.bash_profile.acs -r
make all
```

The corresponding tasks are:

```
- name: copying script with environment variables and make all
  template:
    src: scriptmakeall.j2
    dest: /home/{{ user }}/ACS/ExtProd/INSTALL/scriptmakeall.sh

- name: executing script with environment variables and make all
  shell:
    chdir: /home/{{ user }}/ACS/ExtProd/INSTALL
    cmd: bash scriptmakeall.sh
```

```
    register: logall

  - name: showing stdout make all
    debug:
      msg: "{{ logall.stdout }}"
```

Something very similar and with the same characteristics happens in the successive actions but, in this case, through the template or script `scriptmakebuild.j2` and adding another attribute (`ignore_errors`) when executing the desired script due to a bug not needed be concerned about. The fundamental application of this script is to call to the `make build` command to compile ACS.

The content of the mentioned script is:

```
#!/bin/bash
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
source /home/{{user}}/.acs/.bash_profile.acs -r
export INTROOT=/alma/ACS-2020DEC/ACSSW
make build
```

The corresponding tasks are:

```
- name: copying script with environment variables and make build
  template:
    src: scriptmakebuild.j2
    dest: /home/{{ user }}/ACS/scriptmakebuild.sh

- name: executing script with environment variables and make build
  shell:
    chdir: /home/{{ user }}/ACS
    cmd: bash scriptmakebuild.sh
  ignore_errors: yes
  register: logbuild

- name: showing stdout make build
  debug:
    msg: "{{ logbuild.stdout }}"
```

Next, the directory where the libraries and executables of the development environment will be stored is created by way of simulating the default graphical interface.

```
- name: simulating getTemplate
  shell:
    chdir: /alma/ACS-2020DEC/ACSSW/bin
    cmd: ./getTemplateForDirectory INTROOT /home/{{ user }}/introot_2020DEC
```

As it was done initially in this role with the redirection of a symbolic link, at this point the same thing takes place. The `introot` symbolic link must signal to the directory generated in the preceding task.

```
- name: removing the introot symbolic link
  file:
    path: /home/{{ user }}/introot
    state: absent

- name: creating the introot symbolic link
  file:
    src: /home/{{ user }}/introot_2020DEC
    dest: /home/{{ user }}/introot
    state: link
```

Then, the development environment strictly speaking where the ACS components will be written is downloaded from a repository using the `subversion` module.

```
- name: creating the gtar symbolic link
- name: downloading the development environment
  subversion:
    repo: http://hercules.oan.es/svn/trunk/aries21_2015.4
    dest: /home/{{ user }}/aries21_2020DEC
  ignore_errors: yes
```

Afterwards, several lines are inserted in the system's `.bashrc` file to establish some crucial environment variables to avoid errors during the code compilation.

```
- name: creating the gtar symbolic link
- name: initializing environment variables
  blockinfile:
    path: /home/{{ user }}/.bashrc
    insertbefore: 'export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")'
    block: |
      export INTROOT=$HOME/introot
      export ACS_CDB=$HOME/aries21_2020DEC/OANCDB
      export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
      source $HOME/.acs/.bash_profile.acs -r
      export CLASSPATH=/usr/share/java/jfreechart.jar:"${CLASSPATH}"
```

To allow the virtual environment of Python to import the system Python libraries a document should be made for both available versions of Python with the pertinent content, as appropriate.

```
- name: tweaks for importing system libraries
  blockinfile:
    path: /alma/ACS-2020DEC/pyenv/versions/2.7.16/lib/python2.7/site-packages/dist-packages.pth
    block: |
      /usr/lib/python2.7
      /usr/lib/python2.7/plat-x86_64-linux-gnu
      /usr/lib/python2.7/lib-tk
      /usr/lib/python2.7/lib-old
      /usr/lib/python2.7/lib-dynload
      /usr/local/lib/python2.7/dist-packages
      /usr/lib/python2.7/dist-packages
    create: yes

- name: tweaks for importing system libraries
  blockinfile:
    path: /alma/ACS-2020DEC/pyenv/versions/3.6.9/lib/python3.6/site-packages/dist-packages.pth
    block: |
      /usr/lib/python37.zip
      /usr/lib/python3.7
      /usr/lib/python3.7/lib-dynload
      /usr/local/lib/python3.7/dist-packages
      /usr/lib/python3/dist-packages
    create: yes
```

At last, it is necessary to install the GPIB Linux module in order to prevent mistakes in the control system during the compilation of some of the components. For this purpose, first, the installation package must be downloaded and decompressed in the chosen directory that should be created.

```
- name: creating directory for GPIB Linux module
  file:
    path: /home/{{ user }}/ext_packages
```

```
    state: directory

- name: downloading GPIB Linux module installation package
  get_url:
    url: https://sourceforge.net/projects/linux-gpib/files/latest/download
    dest: /home/{{ user }}/ext_packages/linux-gpib.tar.gz

- name: decompressing downloaded file
  unarchive:
    src: /home/{{ user }}/ext_packages/linux-gpib.tar.gz
    dest: /home/{{ user }}/ext_packages
    remote_src: yes
```

The files that the previous downloaded package contains are decompressed, too.

```
- name: listing ext_packages directory
  shell:
    chdir: /home/{{ user }}/ext_packages
    cmd: ls -d */
  register: lsout

- name: listing decompressed directory
  shell:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}
    cmd: ls
  register: lsout2

- name: decompressing a file in the previous directory
  unarchive:
    src: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout2.stdout_lines.0 }}
    dest: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}
    remote_src: yes

- name: decompressing another file in the previous directory
  unarchive:
    src: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout2.stdout_lines.1 }}
    dest: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}
    remote_src: yes
```

Finally, some required tasks are executed in the two prior directories decompressed for the right installation of the GPIB Linux module, the last of them in each case with `root` privileges out of necessity.

```
- name: listing the first decompressed directory
  shell:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}
    cmd: ls
  register: lsout3

- name: executing make in kernel directory
  make:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout3.stdout_lines.0 }}

- name: executing make install in kernel kernel
  make:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout3.stdout_lines.0 }}
    target: install
  become: yes
  become_method: su

- name: executing ./configure in user directory
  shell:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout3.stdout_lines.2 }}
    cmd: ./configure --sysconfdir=/etc
```

```
- name: executing make in user directory
  make:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout3.stdout_lines.2 }}

- name: executing make install in user directory
  make:
    chdir: /home/{{ user }}/ext_packages/{{ lsout.stdout_lines.0 }}/{{ lsout3.stdout_lines.2 }}
    target: install
  become: yes
  become_method: su
```

4.3 Execution process

First of all, the repository where the Ansible files are placed is cloned by:

```
git clone http://git.oan.es/acs-control-systems/acs-2020dec-ansible.git
```

Then, the SSH key `ansible_key` needed is generated via the following commands in the `keys` directory placed in the same location as the playbook:

```
cd acs-2020dec-ansible
mkdir keys
ssh-keygen -q -N acs-ansible-passphrase -f keys/ansible_key
```

Later, the `.pub` SSH key file must be copied in the `authorized_keys` document of the remote machines included in the inventory file situated at `/etc/ansible/hosts`.

In order to run the mentioned playbook, it is invoked the next command indicating the used SSH key in the `-private-key` option:

```
ansible-playbook --private-key=keys/ansible_key --ask-become-pass acs.yml
```

The obtained output with the related password request at the beginning because of the `--ask-become-pass` command option specified more or less is:

```
name
SUDO password:

PLAY [root role]    ***************************************************************************

TASK [Gathering Facts]    ***********************************************************************
ok: [172.16.10.1]

TASK [root : adding variables]    *****************************************************************
ok: [172.16.10.1]

TASK [root : adding the needed repositories]    ****************************************************
changed: [172.16.10.1] => (item=deb http://deb.debian.org/debian/ buster main)
changed: [172.16.10.1] => (item=deb-src http://deb.debian.org/debian/ buster main)
ok: [172.16.10.1] => (item=deb http://security.debian.org/debian-security buster/updates main)
ok: [172.16.10.1] => (item=deb-src http://security.debian.org/debian-security buster/updates main)
changed: [172.16.10.1] => (item=deb http://deb.debian.org/debian/ buster-updates main)
changed: [172.16.10.1] => (item=deb-src http://deb.debian.org/debian/ buster-updates main)

TASK [root : installing the needed packages]     ***********************************************
changed: [172.16.10.1]
```

```
TASK [root : creating the /alma directory]     ********************************************************
changed: [172.16.10.1]

TASK [root : creating the gtar symbolic link]     ********************************************************
changed: [172.16.10.1]

PLAY [user role]    ***************************************************************************************

TASK [Gathering Facts]    ***********************************************************************************
ok: [172.16.10.1]

TASK [user : cloning the ACS repository]    ********************************************************
changed: [172.16.10.1]

TASK [user : removing the ACS symbolic link]    ********************************************************
changed: [172.16.10.1]

TASK [user : creating the ACS symbolic link]    ********************************************************
changed: [172.16.10.1]

TASK [user : correcting mistakes]    ********************************************************************
changed: [172.16.10.1]

TASK [user : correcting mistakes]    ********************************************************************
changed: [172.16.10.1]

TASK [user : correcting mistakes]    ********************************************************************
changed: [172.16.10.1]

TASK [user : adding package]    ************************************************************************
changed: [172.16.10.1]

TASK [user : correcting mistakes]    ********************************************************************
changed: [172.16.10.1]

TASK [user : adding package]    ************************************************************************
changed: [172.16.10.1]

TASK [user : creating the .acs directory]    ********************************************************
changed: [172.16.10.1]

TASK [user : copying .bash_profile.acs]    *************************************************************
changed: [172.16.10.1]

TASK [user : initializing environment variables]    ****************************************************
changed: [172.16.10.1]

TASK [user : copying script with environment variables and make all]    ******************************
changed: [172.16.10.1]

TASK [user : executing script with environment variables and make all]    *****************************
changed: [172.16.10.1]

TASK [user : showing stdout make all]    ***************************************************************
(similar to the non-automatic installation but without format)

TASK [user : copying script with environment variables and make build]    *****************************
changed: [172.16.10.1]

TASK [user : executing script with environment variables and make build]    ****************************
(similar to the non-automatic installation but without format)

TASK [user : showing stdout make build]    ************************************************************
(similar to the non-automatic installation but without format)

TASK [user : simulating getTemplate]    ***************************************************************
changed: [172.16.10.1]
```

```
TASK [user : removing the introot symbolic link]    ************************************************
changed: [172.16.10.1]

TASK [user : creating the introot symbolic link]    ************************************************
changed: [172.16.10.1]

TASK [user : downloading the development environment]    ******************************************
changed: [172.16.10.1]

TASK [user : initializing environment variables]    ************************************************
changed: [172.16.10.1]

TASK [user : tweaks for importing system libraries]    ********************************************
changed: [172.16.10.1]

TASK [user : tweaks for importing system libraries]    ********************************************
changed: [172.16.10.1]

TASK [user : creating directory for GPIB Linux module]    ****************************************
changed: [172.16.10.1]

TASK [user : downloading GPIB Linux module installation package]    ********************************
changed: [172.16.10.1]

TASK [user : decompressing downloaded file]    ***********************************************************
changed: [172.16.10.1]

TASK [user : listing ext_packages directory]    ***********************************************************
changed: [172.16.10.1]

TASK [user : listing decompressed directory]    ***********************************************************
changed: [172.16.10.1]

TASK [user : decompressing a file in the previous directory]    ***********************************
changed: [172.16.10.1]

TASK [user : decompressing another file in the previous directory]    *******************************
changed: [172.16.10.1]

TASK [user : listing the first decompressed directory]    ****************************************
changed: [172.16.10.1]

TASK [user : executing make in kernel directory]    ************************************************
changed: [172.16.10.1]

TASK [user : executing make install in kernel kernel]    *****************************************
changed: [172.16.10.1]

TASK [user : executing ./configure in user directory]    *****************************************
changed: [172.16.10.1]

TASK [user : executing make in user directory]    ************************************************
changed: [172.16.10.1]

TASK [user : executing make install in user directory]    *****************************************
changed: [172.16.10.1]

PLAY RECAP ***********************************************************************************************
172.16.10.1                : ok=45   changed=38   unreachable=0    failed=0
```

References

[1] Beltran, F. J. Barbas, L. de Vicente, P. Nyalesund 13.2m control system: Installation and Basics. OAN Technical Report 2016-17.

[2] de Vicente, P. Bolaño, R. Selección de herramientas de software para los programas de control del radiotelescopio de 40M. OAN Technical Report 2003-4.

[3] de Vicente, P. Bolaño, R. First impressions using ACS. Installing and using ACS from scratch. OAN Technical Report 2004-1.

[4] de Vicente, P. Bolaño, R. Barbas, L. Primera prueba con el sistema de control del radiotelescopio de 40m. Observaciones. OAN Technical Report 2007-8.

[5] de Vicente, P. ARIES21, the control command line shell for the 40m radiotelescope. OAN Technical Report 2008-9.

[6] "Ansible Documentation", Ansible, 2021. [Online]. Available on: https://docs.ansible.com. [Accessed: 6th April 2021].