

Tool Administrators

Tool Administrator responsibilities and procedures.

Installing Tool Dependencies

Occasionally additional software needs to be installed to in the tool execution environment for a tool. Please following these instructions in the following order. Keep in mind that this environment is common for all tools.

Install a Debian package in the tool execution environment template

Previously you should check that the software to be installed is available for the template operating system at <https://packages.debian.org/> .

As root enter the 'chroot' template environment (Debian 7 as of this writing, but could be Debian 8 etc. modify the commands as necessary).

```
sudo chroot /var/lib/vz/template/debian-7.0-amd64-maxwell/
```

Check the release version of the template operating system.

```
# cat /etc/issue
```

```
Debian GNU/Linux 7 n l # as of this writing.
```

Always update packages before installing new ones.

```
# apt-get update  
# apt-get upgrade
```

Install the new package

```
# apt-get install <packagename>
```

Exit the chroot environment.

```
# exit
```

Manually install software in the 'use' infrastructure

Typically we create and run Hapi scripts to configure, compile, and install software in the tool execution environment. This is especially important when multiple versions of the same software must be available to individual tools. For example, Tool A may require R version 3.2.5 while Tool B may require R version 3.5.0 neither of which could use any other version of R. "Use" is very useful in this case among other to be able to load specific software versions on a per tool basis.

All operations for manually installing dependent software for tools must be done by the apps user from a workspace terminal.

As the apps user clone the Hapi repo into the apps home directory:

```
git clone https://github.com/hubzero/hapi.git
```

Hapi scripts are shell scripts and if one exists for software that you need, just run it! Feel free too add a new Hapi script you created to the repo by submitted a pull request. We add scripts occasionally too.

If a script doesn't exist for the software you need, copy one of the Hapi script and then modify it. Hapi scripts make you life much easier by downloading, configuring, compiling, installing and even adding the require 'use' environ.d file to the appropriate locations.

All tool dependencies are installed by the apps user in /apps/share/debian7 or /apps/share64/debian7 and should be owned by the apps user and groups. All files must be readable by everyone and all directories must be searchable by everyone. No files or directories should be writable by everyone (seriously don't do this)

* For a full manual run "man use" from a tool session terminal.

NAME

use, unuse - adjust the shell environment

SYNOPSIS

```
use [options]... [ENVIRONMENT]
unuse [options]... [ENVIRONMENT]
```

DESCRIPTION

The use command incorporates the specified ENVIRONMENT to the current shell. The unuse command removes it. It optionally records the selection persistently so that subsequent shells will use the ENVIRONMENT. These commands are independent of the shell being run.

An ENVIRONMENT is specified by a configuration file of the same name as found in one of the configuration directories. The ENVIRON_CONFIG_DIRS environment variable specifies a list of directories in which to search for configurations. Each configured ENVIRONMENT specifies a environment variables to set or prepend, shell variables to set, and shell aliases to set.

Some environments are configured to conflict with others. The use command will ask if conflicting ENVIRONMENT should be replaced.

With no arguments, the use and unuse commands will print a synopsis of options and lists all available environments.

- h print available help for a named ENVIRONMENT.
- e environment only. Do not ask about preserving the selection.
- p modify the environment and preserve selection. Do not ask about preserving the selection.
- k keep any conflicting environment. Do not ask about replacing it.
- r replace any conflicting environment without asking.
- x quietly ignore the command if the named ENVIRONMENT cannot be found.

MAKING IT WORK

The following command will describe an environment named xyz:

```
use -h xyz
```

The following command will incorporate the xyz environment preserving the environment for future shell invocations. It will also not override any conflicting environments:

```
use -p -k xyz
```

The following command will remove the xyz environment but retain its use for future sessions:

unuse -e xyz

INTERNAL OPERATION

use and unuse are actually implemented as shell functions (or as aliases in the case of csh derivatives). The functions pass their arguments to the /etc/environ script which determines the commands that the shell should execute to satisfy the new environment configuration. The script prints these commands, the shell function receives them and evals them.

ENVIRONMENT CONFIGURATION FILES

Configuration files are interpreted shell scripts. Several predefined functions are available to make the the process automatic.

alias NAME "Replacement"

Set a command alias in the shell.

conflict VARNAME

Define an environment variable to indicate that a type of an ENVIRONMENT is in use. All conflicting ENVIRONMENT configurations should specify the same conflict. An ENVIRONMENT configuration may specify multiple conflicts.

desc "A short description..."

A short description of the ENVIRONMENT.

help "A lengthy description..."

A long description of the ENVIRONMENT and how to use it. This description will be formatted when printed.

prepend VARNAME ADDITION

Prepend ADDITION to the environment variable VARNAME separated with a colon.

setenv VARNAME REPLACEMENT

Set or replace the environment variable VARNAME with REPLACEMENT.

shellset VARNAME REPLACEMENT

Set or replace the shell variable VARNAME with REPLACEMENT.

Adding additional packages to Jupyter Notebooks

Google the desired package (python, R packages, not OS packages) and review the installation instructions. They might recommend a different conda repository than the default.

Be careful if conda says it wants to downgrade packages. If it is a minor downgrade, it is probably OK. Do not proceed if many packages must be downgraded or critical packages are to be downgraded. Remember that changes to the Anaconda environment will affect all tools using the environment and a downgrade could cause tools not to function (if a feature is no longer available, for example).

- Start a Workspace
- From the terminal in the Workspace switch to the Apps user (your account must be a member of the apps group in the CMS).
 - `sudo su - apps`
- Load the Anaconda environment that you will to modify (there may be multiple Anaconda environments available. execute the command "use" from the terminal for a list)
 - `use anaconda3-5.1`
- Install the desired package via 'conda'. 'conda' is preferred over 'pip'.
 - `conda install <pkgname>`
 - `conda install -c conda-forge <pkgname>`
- The installation may take a few minutes
- **Important! Fix any world writable files by doing:**
 - `chmod -R o-w /apps/share64/debian7/anaconda`

If the desired package isn't available in conda or there are issues, you can try pip. Use the following command.

```
pip install -U --upgrade-strategy only-if-needed <pkgname>
```

Creating a separate anaconda environment.

```
#load the anaconda3-5.1 environment
conda create -n <environame> # create the <environame> environment
source activate <environame> # enter the <environame> environment
conda install <packagename> # install <packagename> and its dependencies
conda deactivate # exit the <environame> environment
```

Create environ.d file for <environame> environment. Users must load the anaconda3-5.1 environment first and the <environame> environment.

Reference:

<https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

Updating hublib

```
sudo su - apps
use anaconda3-5.1
pip install -U hublib
```

White list of directories through the CMS

[Download \(pdf\)](#)

Any super-user can whitelist a directory via the /administrator interface. Follow these steps to complete this request:

1. Navigate to the **/administrator** interface and login
2. Hover over **Components** then click on **Tools**
3. Click the **Options** button and under **Directory Parameter Whitelist**, add a "," then the directory
 1. For example: , /home
4. Click **Save & Close**
5. Make sure that you complete this task on all the requested machines (i.e. production, stage, dev, qa, etc.)