

3.2 Entornos virtuales de Python

[Instalaciones de Python en entornos de Analítica Avanzada](#)

[Python del Sistema](#)

[Entornos Virtuales de Python](#)

[Concepto](#)

[Alternativas](#)

[Buenas prácticas](#)

[Gestión de entornos virtuales con conda](#)

[Introducción a conda](#)

[Ventajas de utilizar conda](#)

[Seguridad](#)

[Resolución automática de entornos](#)

[Rendimiento](#)

[Simplicidad](#)

[Instalación de conda](#)

[Python 2 vs Python 3](#)

[Anaconda vs Miniconda](#)

[Guía de instalación](#)

[Comandos básicos de conda](#)

[Obtención de información de la instalación](#)

[Creación de un nuevo entorno virtual](#)

[Obtención de entornos existentes](#)

[Activación de los entornos](#)

[Instalación de Python en el entorno](#)

[Instalación de librerías en un entorno](#)

[Eliminar librerías instaladas en el entorno](#)

[Obtención de librerías instaladas en entorno](#)

[Obtención de información de una librería](#)

[Eliminación de un entorno junto con sus librerías](#)

[Clonar entornos](#)

[Distribución de entornos](#)

[Con archivo de configuración de entornos de conda \(.yaml\)](#)

[Comprimiendo el entorno con la herramienta conda-pack](#)

[Localización del ejecutable Python](#)

[Gestión de entornos virtuales con virtualenv](#)

[Test de conocimientos](#)

Instalaciones de Python en entornos de Analítica Avanzada

Python del Sistema

Como se ha explicado en el tema anterior, Python es el lenguaje de referencia en tareas analíticas, comprendiendo estas principalmente el tratamiento de datos en memoria y el entrenamiento y puesta en producción de modelos analíticos y sistemas de Inteligencia Artificial.

Python se encuentra ya instalado por defecto en sistemas basados en Unix, como OSX y Linux. Esta instalación de Python se ubica normalmente en las rutas para la mayoría de distribuciones de Linux:

- Python 2: `/usr/bin/python`
- Python 3: `/usr/bin/python3`

A este Python se le denomina por convención la **instalación de Python del Sistema**. Esta instalación de Python es utilizada por una gran cantidad de aplicaciones, programas, procesos en segundo plano, etc., que se ejecutan en el sistema operativo.

En el caso de sistemas con Windows, Python no se encuentra instalado por defecto en el sistema. No es habitual encontrar software desarrollado para Windows en Python ni el uso de Windows para realizar tareas de analítica.

Se recomienda que la **instalación de Python del Sistema** no sea modificada por el usuario en ningún caso, lo que podría provocar que procesos del sistema y aplicaciones dejen de funcionar de forma adecuada.

Entornos Virtuales de Python

Concepto

En entornos de trabajo analíticos, es común la experimentación con diferentes tipos de modelos y técnicas, lo que requiere la utilización de multitud de librerías. Muchas librerías no son compatibles entre sí, y su instalación podría ‘romper’ la instalación de Python y quedar esta totalmente inusable si hay alguna colisión entre librerías.

Si la instalación de estas librerías se realiza en el Python del sistema, esto podría provocar que multitud de aplicaciones no funcionen correctamente, llegando incluso a tener que formatear el equipo y reinstalar el sistema desde cero.

Para solventar este problema, existen los llamados Entornos Virtuales. Estos entornos no son más que instalaciones de Python que cumplen con las siguientes características:

- Están aisladas del Python del Sistema: su instalación se realiza habitualmente en otro directorio y no tiene ninguna relación con el Python ya instalado por defecto en el sistema, por lo que la instalación de librerías y la experimentación con dicho entorno no provocará daños en el Python del Sistema.
- Incluyen algún tipo de gestor de entornos: dicho gestor es un software (normalmente en forma de aplicación utilizada mediante interfaz línea de comandos) que facilita la creación y la destrucción de dichos entornos de forma sencilla y automática.

Alternativas

Hay diversas alternativas para utilizar entornos virtuales. Las dos más utilizadas son:

- conda: es un gestor creado para la distribución de Python de Anaconda Inc. Esta distribución de Python es una de las más utilizadas en Analítica por las ventajas que ofrece.
- virtualenv/venv: es una alternativa para la creación de entornos virtuales que ya ha sido integrado en la librería estándar de Python desde la versión 3.3 del mismo. Tiene algunas ventajas y desventajas respecto a conda que se especificarán más adelante.

Buenas prácticas

Tanto para la experimentación, como para la puesta en producción de modelos de Inteligencia Artificial sobre Python, se recomienda nunca depender de la instalación de Python del Sistema. En dicho caso, se incurrirá en el riesgo de instalar librerías incompatibles entre sí, o que modifican dicha instalación de forma que esta deje de funcionar correctamente. La opción más simple es utilizar el gestor de entornos virtuales **conda**.

Gestión de entornos virtuales con conda

Introducción a conda

Conda es un potente administrador de paquetes y administrador de entornos virtuales disponible para Linux, Windows y OSX, que se puede utilizar mediante:

- Línea de comandos: la forma recomendada de trabajar con conda.
- Interfaz gráfica: llamada Anaconda Navigator. Permite, de forma gráfica, crear entornos, instalar librerías, destruir entornos, etc. No es tan robusta como la interfaz de línea de comandos, aunque es más sencilla para usuarios que no están familiarizados con interfaces CLI (command line interface).

Ventajas de utilizar conda

Seguridad

A pesar de que el administrador de paquetes por defecto de Python, llamado pip, es excelente, adolece de ciertos problemas, siendo el más importante de ellos la seguridad de los paquetes descargados desde el mismo. A pesar del excelente trabajo de los encargados del Python Package Index, es imposible controlar al detalle el código subido por los usuarios puesto que esta subida de código está abierta a todo el mundo sin ningún tipo de control.

La brecha de seguridad más común es la suplantación de los paquetes. Por ejemplo, es muy sencillo cometer una errata en la instalación de un paquete (requests vs request) y que el paquete instalado tenga un comportamiento aparentemente similar al original, pero que incluya código malicioso.

En el caso de conda el control de los paquetes incluidos, sobre todo en el repositorio principal, es más exhaustivo (aunque no perfecto), siendo mucho más difícil la subida de software malicioso. Es por ello que en la instalación de Python en infraestructura de clientes y en procesos en producción, es muy recomendable utilizar conda sobre pip.

Resolución automática de entornos

La resolución de entornos consiste en el proceso de comprobación de dependencias entre las diferentes librerías que van a ser instaladas en el mismo hasta encontrar una configuración de versiones de las mismas que ‘satisfaga’ los requerimientos de todas ellas a la vez.

El gestor de entornos por defecto de Python, pip, realiza una resolución básica de los entornos, pero conda realiza una resolución completa a costa de la velocidad de instalación. La resolución que realiza conda es muy exhaustiva, comprobando las versiones de las librerías navegando por los requerimientos de las mismas de forma recursiva, resultando en entornos bastante consistentes que es complicado romper.

Rendimiento

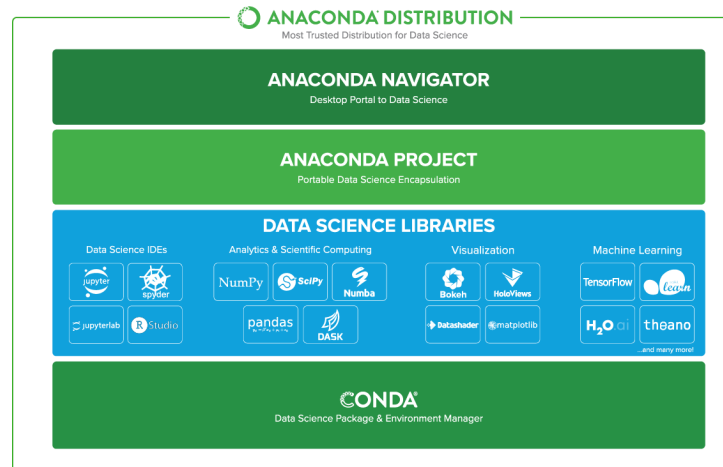
Existen algunas librerías cuya base principal de código no es Python, sino otro lenguaje de programación de más bajo nivel (más cercano a las operaciones lógicas del propio procesador) pero más rápido. Un ejemplo es TensorFlow, escrito en C++ principalmente.

La forma de instalar este tipo de librerías varía dependiendo de si son instaladas utilizando pip o conda. Se han llegado a observar [diferencias](#) de rendimiento muy importantes entre las instalaciones desde conda o pip, por lo que se recomienda, siempre que sea posible, instalar dichas librerías utilizando conda y únicamente depender de pip para la instalación en el caso de que las librerías no estén disponibles en conda.

Simplicidad

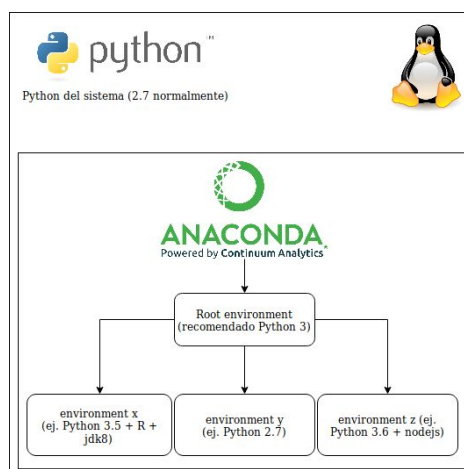
La funcionalidad de instalación de paquetes del gestor de entornos conda permite instalar más cosas aparte de librerías de Python. Se puede instalar Node.js, OpenJDK (Java), CUDA, etc. Incluso es capaz de gestionar la instalación de librerías complejas como la versión de Tensorflow que corre sobre tarjetas gráficas (GPU) con mínima intervención del usuario, lo que ahorra un tiempo valioso en instalación y configuración.

En resumen, instalaciones que por su complejidad requerirían seguir algún tipo de tutorial para realizarlas de forma correcta y en consecuencia de una gran cantidad de tiempo, son realizadas de forma automática por conda, lo que puede reducir el tiempo de instalación de horas a minutos. La gran cantidad de requerimientos que tiene Tensorflow en su versión GPU (CUDA, cuDNN, etc.) no son de trivial instalación, pero son automáticamente instalados por conda.



Instalación de conda

Instalar conda es totalmente compatible con virtualenv, y no afecta para nada a la instalación de Python del Sistema, quedando de forma esquematizada una instalación de conda con sus entornos en un sistema Linux de la siguiente forma:



Hay que tener en cuenta algunas consideraciones importantes a la hora de realizar la instalación de conda y tomar algunas decisiones.

Python 2 vs Python 3

Este ha sido un debate abierto durante mucho tiempo en la industria. Actualmente existen millones de líneas de Python 2 en producción en todo el mundo.

Muchas empresas siguen teniendo Python 2 como su lenguaje vehicular en analítica. Es posible (no sin cierto esfuerzo) crear código compatible con Python 2 y Python 3 aunque hoy en día la **opción prioritaria debe ser siempre Python 3**.

Es importante notar que Python 2 perderá cualquier tipo de soporte (incluyendo actualizaciones de seguridad) a partir de 2020. La mayoría de librerías ya han dado el salto a Python 3 y todos los nuevos desarrollos realizados en los últimos años son en Python 3.

Las principales diferencias entre ambas versiones son las siguientes:

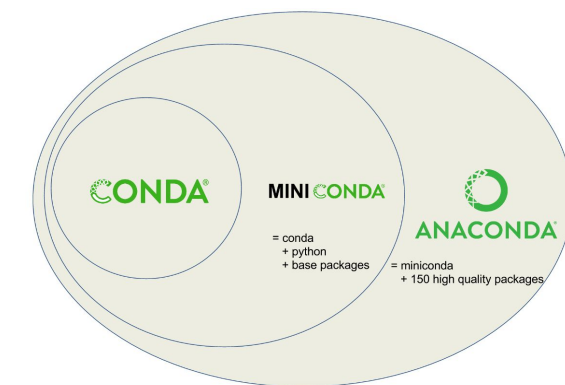
- El tipo string en Python 3 es unicode por defecto.
- El uso de generadores en Python 3 (`dict.keys()` o `dict.values()` cambian su comportamiento).
- El operación de división devuelve float en Python 3, mientras que en la versión 2 únicamente la parte entera de la división.
- `print` es una función en Python 3.

Anaconda vs Miniconda

Existen 2 formas de realizar la instalación de la distribución de Python de Anaconda:

- Anaconda: incluye ~500Mb de librerías preinstaladas (pandas, scikit-learn, etc.).
- Miniconda: es una versión 'ligera' de Anaconda con las mismas funcionalidades y que únicamente instala el gestor de entornos virtuales. Esta es la opción recomendada para infraestructura cloud o equipos con almacenamiento limitado (portátiles con discos duros SSD).

Nota importante: el hecho de que Anaconda ya incluye 500Mb de librerías preinstaladas no quiere decir que estas librerías no estén disponibles en Miniconda, simplemente que al solicitar su instalación, estas librerías no necesitarán ser descargadas, estando ya disponibles (cacheadas) en el sistema. Las librerías disponibles en ambas instalaciones son **las mismas**.



Guía de instalación

Para instalar Anaconda (en su versión ligera Miniconda) se ha de acceder primero a la página de descargas en la siguiente URL:

- <https://docs.conda.io/en/latest/miniconda.html>

Se seleccionará la instalación de 64 bits correspondiente al sistema operativo (Windows, OSX o Linux). En este curso se recomienda realizar la instalación en Linux. En el caso de no tener disponible una máquina con Linux (se

puede realizar la instalación sin problema en la máquina virtual de Cloudera) se recomienda crear una máquina virtual con Ubuntu 18.04 usando VirtualBox. En el caso de OSX se puede elegir entre un instalador basado en .pkg (formato de distribución de software propio de Mac) o el instalador .sh (script de instalación con Bash). Se recomienda utilizar la segunda opción (Bash). En el caso de Windows únicamente hay que seguir las instrucciones del ejecutable de instalación (.exe). Se recomienda tener precaución con lo siguiente:

- En el caso de Linux y OSX es importante añadir conda al PATH del sistema, de lo contrario conda no estará disponible al abrir una nueva terminal.
- En el caso de Windows, no es necesario añadir nada al PATH, ya que existe una aplicación (Anaconda Prompt) que no es más que una terminal de Windows con todo lo necesario ya inicializado (conda, python, etc.).

Para realizar la instalación en Linux y OSX, tras descargar el script (.sh) es necesario abrir una terminal en la carpeta de dicha instalación y ejecutarlo con bash, seguir la instalación y aceptar los términos y condiciones. Al finalizar la instalación es necesario aceptar que se conda se añada al PATH.



```
~$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
~$ bash Miniconda3-latest-Linux-x86_64.sh
```

Una vez se ha realizado la instalación, se comprobará que la instalación es correcta escribiendo conda en una terminal y observando que el output es similar al siguiente:

```
(base) david@david-workstation:~$ conda
usage: conda [-h] [-V] command ...
```

```
conda is a tool for managing and deploying applications, environments and packages.
```

```
Options:
```

```
positional arguments:
```

```
command
```

clean	Remove unused packages and caches.
config	Modify configuration values in .condarc. This is modeled after the git config command. Writes to the user .condarc file (/home/david/.condarc) by default.
create	Create a new conda environment from a list of specified packages.
help	Displays a list of available conda commands and their help strings.
info	Display information about current conda install.
init	Initialize conda for shell interaction. [Experimental]
install	Installs a list of packages into a specified conda environment.
list	List linked packages in a conda environment.
package	Low-level conda package utility. (EXPERIMENTAL)

```

remove      Remove a list of packages from a specified conda environment.
uninstall   Alias for conda remove.
run         Run an executable in a conda environment. [Experimental]
search      Search for packages and display associated information. The
            input is a MatchSpec, a query language for conda packages.
            See examples below.
update      Updates conda packages to the latest compatible version.
upgrade     Alias for conda update.

```

optional arguments:

```

-h, --help  Show this help message and exit.
-V, --version Show the conda version number and exit.

```

conda commands available from other packages:

```

build
convert
debug
develop
env
index
inspect
metapackage
pack
render
skeleton

```

Si el output es similar al mostrado es similar, la instalación se ha realizado correctamente.

Comandos básicos de conda

Obtención de información de la instalación

Para obtener información de la instalación, se utilizará el comando:

- `conda info`


```
pi@raspberrypi:~ $ conda info

active environment : None
  shell level      : 0
  user config file  : /home/pi/.condarc
populated config files : /home/pi/berryconda3/.condarc
  conda version     : 4.5.11
  conda-build version : not installed
  python version    : 3.6.1.final.0
  base environment   : /home/pi/berryconda3 (writable)
    channel URLs    : https://conda.anaconda.org/rpi/linux-armv7l
                    https://conda.anaconda.org/rpi/noarch
  package cache     : /home/pi/berryconda3/pkg
                    /home/pi/.conda/pkg
  envs directories  : /home/pi/berryconda3/envs
                    /home/pi/.conda/envs
    platform       : linux-armv7l
    user-agent      : conda/4.5.11 requests/2.18.1 CPython/3.6.1 Linux/4.14.94-v7+ raspbian/9
glibc/2.24
  UID:GID          : 1000:1000
  netrc file       : None
  offline mode     : False
```

Creación de un nuevo entorno virtual

Para crear un entorno virtual (se recomienda la creación de un entorno virtual por proyecto) se utiliza el comando:

- `conda create -n <nombre entorno>`

El nombre del entorno debe ser único y simple, ya que se utiliza activamente en la gestión del mismo (activar, desactivar, etc.).

```
~$ conda create -n <nombre del entorno>
```

Obtención de entornos existentes

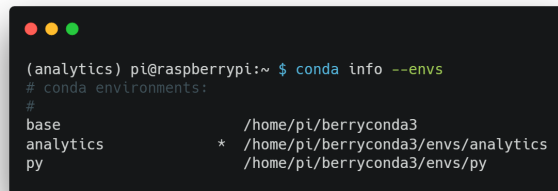
Obtener los entornos instalados es muy útil para:

- Cuando se trabaja con múltiples entornos y no se recuerda exactamente el nombre de aquel con el que se desea interactuar.
- Se quiere conocer el entorno actualmente activo.
- Se quiere obtener la localización (ruta física) de cada uno de los entornos.

El comando que permite obtener este tipo de información es el siguiente:

- `conda info --envs`

El comando es realmente una modificación del anterior (info) con un flag (`--envs`) especificando que la información recuperada ha de ser necesariamente sobre los entornos existentes.



```
(analytics) pi@raspberrypi:~ $ conda info --envs
# conda environments:
#
base                  /home/pi/berryconda3
analytics             * /home/pi/berryconda3/envs/analytics
py                   /home/pi/berryconda3/envs/py
```

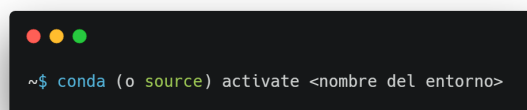
Activación de los entornos

La activación de los entornos es el paso previo que hay que realizar siempre que se vayan a realizar operaciones sobre/con el mismo. Con el entorno activado se pueden instalar librerías en el mismo, así como acceder a aquellas librerías y aplicaciones previamente instaladas.

Una vez se ha activado un entorno, aparecerá el nombre del mismo entre paréntesis en el prompt de la terminal, indicando así que el entorno se encuentra activado, y que las acciones a realizar a partir de entonces se realizarán sobre dicho entorno.

La activación del entorno se realiza con el comando:

- `conda activate <nombre entorno>`



```
~$ conda (o source) activate <nombre del entorno>
```

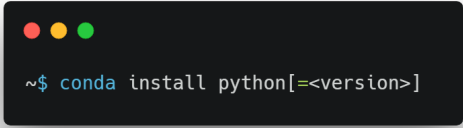
Hay que ser extremadamente cuidadoso al activar y desactivar los entornos, ya que pueden irse acumulando (activar un entorno dentro de otro), llegando a no tener claro en cada momento sobre qué entornos se están realizando las acciones o dónde se encuentran instaladas las librerías que se están utilizando.

Instalación de Python en el entorno

Cuando se crea un entorno y se activa, el entorno se encuentra originalmente vacío, ni siquiera tiene Python instalado, por lo que es necesario realizar la instalación.

La instalación del entorno se realiza con el comando:

- `conda install python`



```
~$ conda install python[=<version>]
```

Se recomienda realizar esta acción nada más crear el entorno. Por defecto (si no se especifica una versión, por defecto instalará la última disponible). Se puede instalar tanto Python 2 como Python 3.

Instalación de librerías en un entorno

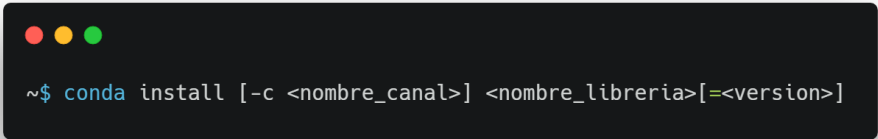
En un entorno de Python es posible instalar librerías. Las librerías son software externo, normalmente open source, que realiza algún tipo de funcionalidad y que se puede utilizar de forma modular en proyectos propios.

La amplia gama de librerías es una de las funcionalidades de Python más atractivas, ya que ahorran mucho tiempo y evitan tener que estar desarrollando constantemente. Se pueden instalar librerías con estas dos modalidades:

- Desde pip: el gestor de librerías por defecto de Python. Se incluye en cualquier instalación de Python.
- Desde conda: el propio gestor de entornos conda. Las librerías se instalan desde unas ubicaciones en la nube llamadas 'canales'. Los canales se pueden ver como repositorios de librerías gestionados por un usuario o entidad. Hay varios canales importantes:
 - conda: el canal por defecto desde el que se instalarán las librerías en caso de que no se especifique ningún otro. Está gestionado por Anaconda Inc. e incorpora las librerías más estables.
 - conda-forge: es el canal donde en general se encontrará cualquier librería de cierta relevancia que no esté en el canal por defecto. También incluye las últimas versiones de las librerías de conda, aunque estas no estén todavía en su versión final.
 - canales propios de usuarios: cualquiera puede crear un canal de conda y subir sus propias librerías para poder instalarlas desde cualquier parte. Hay gran cantidad de empresas que cuentan con su propio canal, donde distribuyen su software.

Las librerías se instalarán utilizando el gestor conda siempre que sea posible. En caso de no estar la librería disponible en conda, se recurrirá a la instalación desde pip. Para instalar una librería desde conda:

- `conda install <nombre librería>`



```
~$ conda install [-c <nombre_canal>] <nombre_libreria>[=<version>]
```

Una vez se ha ordenado la instalación de una librería mediante conda, el gestor mostrará el plan de instalación, al que hay que prestar especial atención para evitar efectos no deseados. El plan de instalación muestra la siguiente información:

- Librerías que van a ser descargadas: las librerías que no se encuentran cacheadas en el sistema y que necesitan ser descargadas (no es muy importante salvo al instalar librerías pesadas o con una conexión a internet lenta).
- Instalación: aquellas librerías que van a ser instaladas en el entorno.
- Actualización: aquellas librerías cuya versión se va a ver modificada a una más reciente.
- Retroceso: aquellas librerías cuya versión se va a ver modificada a una más antigua.

Se ha de prestar especial atención al retroceso de versiones que se puede producir al instalar nuevas librerías. Se muestra a continuación un ejemplo de plan de instalación:

```
(analytics) pi@raspberrypi:~ $ conda install pandas
Solving environment: done

## Package Plan ##

environment location: /home/pi/berryconda3/envs/analytics

added / updated specs:
- pandas

The following packages will be downloaded:
```

package	build	
numpy-1.14.0	py36h73aad69_0	7.4 MB
pytz-2018.5	py_0	193 KB
six-1.11.0	py36_1	20 KB
libgfortran-3.0.0	0	206 KB
openblas-0.2.19	0	2.6 MB
pandas-0.23.4	py36h6b76cdf_0	25.8 MB
python-dateutil-2.6.1	py_0	190 KB
Total:		36.4 MB

```

The following NEW packages will be INSTALLED:

libgfortran: 3.0.0-0
numpy: 1.14.0-py36h73aad69_0
openblas: 0.2.19-0
pandas: 0.23.4-py36h6b76cdf_0
python-dateutil: 2.6.1-py_0
pytz: 2018.5-py_0
six: 1.11.0-py36_1

Proceed ([y]/n)?
```

Tras la instalación, se pide la confirmación del usuario para realizar toda la instalación.

Eliminar librerías instaladas en el entorno

Otra acción muy común es la eliminación de librerías de un entorno cuando estas ya no se necesitan. Para eliminar una librería, habiendo previamente activado el entorno:

- `conda remove <nombre librería>`

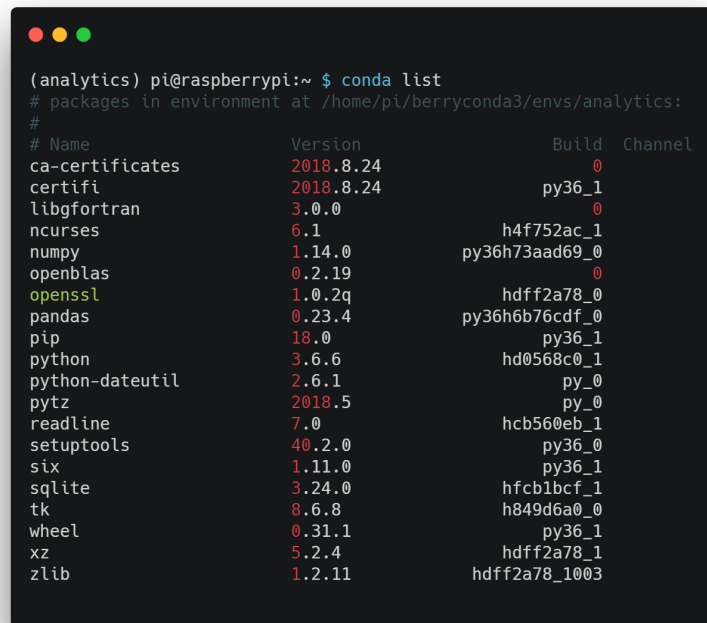
Este tipo de operaciones deben ser hechas con cuidado, ya que al instalar librerías también es posible romper el entorno. Ante fallos repetitivos en un entorno se recomienda crear el entorno desde cero en lugar de instalar y desinstalar librerías.

Obtención de librerías instaladas en entorno

Con el entorno activo, se pueden obtener aquellas librerías que se encuentran actualmente instaladas, así como la versión de cada una y el origen (forma de instalación con pip/conda y el canal de instalación).

Para obtener las librerías instaladas, con el entorno activado:

- `conda list`



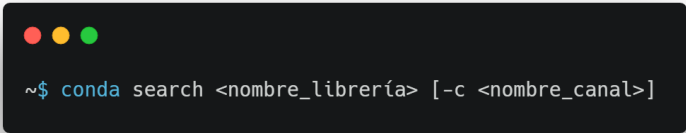
```
(analytics) pi@raspberrypi:~ $ conda list
# packages in environment at /home/pi/berryconda3/envs/analytics:
#
# Name                        Version                        Build      Channel
ca-certificates              2018.8.24                      0
certifi                      2018.8.24                      py36_1
libgfortran                  3.0.0                          0
ncurses                      6.1                            h4f752ac_1
numpy                        1.14.0                        py36h73aad69_0
openblas                    0.2.19                         0
openssl                      1.0.2q                         hdff2a78_0
pandas                      0.23.4                        py36h6b76cdf_0
pip                          18.0                          py36_1
python                      3.6.6                         hd0568c0_1
python-dateutil              2.6.1                         py_0
pytz                        2018.5                         py_0
readline                    7.0                            hcb560eb_1
setuptools                   40.2.0                         py36_0
six                          1.11.0                         py36_1
sqlite                       3.24.0                         hfcblbcf_1
tk                            8.6.8                         h849d6a0_0
wheel                        0.31.1                         py36_1
xz                           5.2.4                         hdff2a78_1
zlib                         1.2.11                         hdff2a78_1003
```

Obtención de información de una librería

Previamente a la instalación de una librería, es una buena práctica revisar las versiones existentes en diversos canales. Un ejemplo es Tensorflow, que tiene versión para diversas arquitecturas computacionales (CPU, GPU, etc.).

Para obtener información sobre una librería:

- `conda search <nombre librería>`

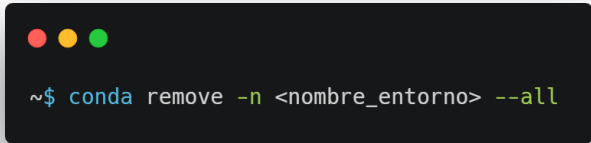


```
~$ conda search <nombre_librería> [-c <nombre_canal>]
```

Eliminación de un entorno junto con sus librerías

Para eliminar completamente un entorno (por falta de uso o por algún error en el mismo) de forma sencilla y sin dejar rastro en el equipo (no hay forma de recuperar el entorno una vez eliminado, siendo necesaria la creación de uno nuevo desde cero):

- `conda remove -n <nombre_entorno> --all`



```
~$ conda remove -n <nombre_entorno> --all
```

Clonar entornos

Para crear un entorno idéntico a uno existente:

- `conda create --name <nombre_clonado> --clone <nombre_original>`

Distribución de entornos


Con archivo de configuración de entornos de conda (.yaml)

A veces es necesario la distribución de entornos creados en un equipo. Hay varias metodologías a la hora de distribuir los entornos, cada una con sus ventajas y sus inconvenientes. La forma de distribuir el entorno depende principalmente del gestor utilizado para su creación. A continuación se explican dos formas de distribuir los entornos utilizando conda.

Es posible definir los requerimientos de un entorno en un archivo .yaml de la siguiente forma, donde se pueden especificar:

- Nombre del entorno: un nombre que lo identifique de forma unívoca.
- Canales: los canales donde se van a buscar las librerías a instalar.
- Dependencias (conda): las librerías a instalar junto con su versión.
- Dependencias (pip): las librerías que se van a instalar desde pip junto con su versión.

Para crear un entorno desde un archivo .yaml, se utilizará el siguiente comando:



```
~$ conda env create -f ruta/al/archivo.yaml
```

Para exportar un entorno activo y crear automáticamente un archivo .yaml desde dicho entorno, se puede utilizar el siguiente comando:


- `conda env export > environment.yaml`

Comprimiendo el entorno con la herramienta conda-pack

Recientemente, existe la opción de comprimir el entorno y poder distribuirlo en cualquier equipo con Linux. Esta forma de trabajar tienes las siguientes ventajas:

- Es posible desplegar el entorno en localizaciones que no tienen conexión a internet.
- El entorno queda congelado tal cual, evitando tener que descargar e instalar las librerías en la nueva localización.

Para poder acceder a esta funcionalidad es necesario instalar la librería conda-pack en el entorno base de conda (el entorno llamado base es aquel que se crea por defecto al instalar conda). A día de hoy, conda-pack debe instalarse desde el canal conda-forge al no encontrarse disponible en el canal principal. Para realizar esta instalación:



```
~$ conda install -c conda-forge conda-pack
```

Una vez se ha instalado conda-pack, es posible empaquetar un entorno, habiendo previamente activado el entorno antes. Se ha de navegar con la terminal hasta el directorio donde se desee crear el archivo del entorno empaquetado. Una vez en la ruta deseada, se utiliza conda-pack para empaquetar el entorno en un archivo comprimido:

```
(analytics) pi@raspberrypi:~ $ conda pack
Collecting packages...
Packing environment at '/home/pi/berryconda3/envs/analytics' to 'analytics.tar.gz'
[#####] | 100% Completed | 2min 10.1s
```

El archivo comprimido creado por la utilidad conda-pack puede ser copiado en cualquier equipo con Linux. Una vez en la localización de destino deseada será necesario:

- Crear una carpeta donde descomprimir el entorno.
- Descomprimir el entorno en dicha carpeta
- Ejecutar el script de activación en el directorio bin.
- Ejecutar la aplicación conda-unpack, que crea todos los links necesarios y ejecuta las acciones para que todo el entorno funcione de forma consistente.


```
~$ mkdir carpeta_entorno
~$ tar xzvf entorno.tar.gz -C carpeta_entorno/
~$ source carpeta_entorno/bin/activate
~$ conda-unpack
```

Localización del ejecutable Python

Cuando se trabaja con entornos virtuales, la localización del ejecutable de Python del entorno es muy importante para:

- Poder ejecutar scripts de Python desde la terminal.
- Poder utilizar algún entorno de desarrollo con funcionalidades avanzadas de introspección (como por ejemplo PyCharm), que analiza las librerías instaladas para autocompletar.

Por defecto, conda (en su versión miniconda para Python 3) se instala en `/home/<nombre de usuario>/miniconda3`. Los entornos con sus respectivas librerías se crean en un directorio con su propio nombre dentro de otro llamado `envs`. La ruta final al ejecutable de Python para un entorno concreto queda entonces así:



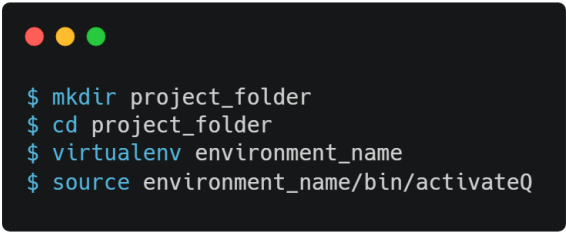
```
~$ <carpeta_anaconda>/envs/<nombre_entorno>/bin/python ruta/al/script.py
```

Gestión de entornos virtuales con virtualenv

Como alternativa a conda, se propone la utilización de virtualenv. Parte de las funcionalidades de virtualenv han sido ya incorporadas a la librería estándar de Python (la librería estándar incluye todo aquello que se distribuye por defecto con Python, como la librerías math, collections, etc.).

Utilizar virtualenv tiene las siguientes ventajas e inconvenientes:

- La instalación es más ligera y ya se encuentra incluida normalmente en los repositorios del sistema operativo. Por ejemplo para realizar la instalación en Ubuntu: `sudo apt install virtualenv`.
- Depende de la versión de Python instalada en el sistema, y no es posible especificar versiones específicas del mismo (por ejemplo, si el Python 3 del sistema es 3.5, únicamente podrán crearse entornos de la versión 3.5 para Python 3).
- Los entornos creados no puede comprimirse y distribuirse de forma sencilla. No existe una funcionalidad similar a la de conda-pack.
- Depende del instalador de paquetes de Python por defecto, pip.



```
$ mkdir project_folder
$ cd project_folder
$ virtualenv environment_name
$ source environment_name/bin/activateQ
```

Test de conocimientos

- 1) Acaba de incorporarse a un departamento de analítica de reciente creación en una empresa. Su superior le pide que analice unos logs de una aplicación web para intentar encontrar patrones de comportamiento de los usuarios. Para ello necesita utilizar una serie de herramientas que no se encuentran originalmente instaladas en su equipo. ¿Cómo procedería?
 - a) Utilizaría el Python 2 del sistema e instalaría las librerías necesarias utilizando pip.

- b) Utilizaría el Python 3 del sistema e instalaría las librerías necesarias utilizando pip.
 - c) Instalaría virtualenv y crearía un entorno virtual sobre el Python 2 del sistema e instalaría las librerías necesarias utilizando pip.
 - d) Instalaría virtualenv y crearía un entorno virtual sobre el Python 3 del sistema e instalaría las librerías necesarias utilizando pip.
 - e) Instalaría conda y crearía un entorno virtual de Python 2 e instalaría las librerías necesarias utilizando conda.
 - f) Instalaría conda y crearía un entorno virtual de Python 3 e instalaría las librerías necesarias utilizando conda.**
- 2) Se encuentra trabajando en un proyecto, para lo que está utilizando un entorno de conda creado por usted mismo para tal efecto. Nota que se producen errores al importar ciertas librerías. ¿Cómo procedería?
- a) Eliminaría el entorno y crearía uno nuevo desde cero.**
 - b) Eliminaría la librería concreta en la que se produce el error y la volvería a instalar.
- 3) Necesita desplegar un entorno de Python en un clúster Big Data que se encuentra aislado de la red (únicamente tiene acceso mediante VPN). ¿Cómo procedería?
- a) Hablaría con el departamento de Sistemas para que ellos realizaran la instalación siguiendo sus instrucciones.
 - b) Se ha de copiar la carpeta de un entorno virtual de conda creado en local en el clúster mediante sftp a alguno de sus nodos.
 - c) Se ha de copiar la carpeta de un entorno virtual de virtualenv creado en local en el clúster mediante sftp a alguno de sus nodos.
 - d) Usaría conda-pack.**
- 4) ¿Qué comando utilizaría para instalar la librería de modelización de Series Temporales [Prophet](#)?
- a) Activaría previamente el entorno y ejecutaría: ``conda install prophet``
 - b) Sobre la instalación de Python del Sistema, ejecutaría: ``pip install prophet``
 - c) Activaría previamente el entorno y ejecutaría: ``conda install -c conda-forge fbprophet``**
- 5) Su usuario del sistema se llama: 'ubuntu', y ha creado un entorno de conda (en su versión miniconda para Python 3) llamado: 'analytics'. Quiere ejecutar un script de python llamado: 'script.py' localizado en su carpeta Desktop. ¿Qué comando utilizaría?
- a) ``/home/ubuntu/miniconda3/envs/analytics/bin/python /home/ubuntu/Desktop/script.py``**
 - b) ``usr/bin/python /home/ubuntu/Desktop/script.py``