CONDA

¿Qué es CONDA?

Conda es un gestor de paquetes y entornos. Es la forma más fácil de manejar la instalación de la mayoría de las herramientas que queremos usar en bioinformática.

Que algo pueda instalarse en Conda, requiere que alguien (el desarrollador u otros) se haya tomado la molestia de hacerlo de esa manera, por lo que no todo está disponible, pero casi todo lo que probablemente queramos usar está. El otro valor de Conda es que facilita la instalación de las cosas y maneja muy bien los diferentes entornos. A veces el Programa A dependerá de una versión específica del Programa B. Pero entonces, el Programa C dependerá de una versión diferente del Programa B, y esto causa problemas. Conda nos permite crear y gestionar fácilmente entornos separados para evitar este tipo de conflictos de versiones, y comprueba automáticamente por nosotros cuando tratamos de instalar algo nuevo. Además, mejora y facilita la reproducibilidad.

Obtener e instalar Conda

Conda se puede obtener mayoritariamente de dos formas: Anaconda y Miniconda. Anaconda es grande y con muchos programas dentro, Miniconda es más ligero, y a partir de ahí, instalas lo necesario.

La página de descarga de Miniconda es: https://conda.io/en/latest/miniconda.html .
Deberemos elegir el adecuado para nuestro sistema operativo.

Instalación en el Work Space de AWS:

instalación de Conda (Miniconda):

creación de carpeta, preferiblemente en /home/USUARIO

```
mkdir -p ~/miniconda3
```

#descargar el instalador

```
wget repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O
~/miniconda3/miniconda.sh
```

#correr el instalador

```
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
```

#eliminar el instalador

```
rm -rf ~/miniconda3/miniconda.sh
```

#ejecutar conda por defecto en el terminal

```
~/miniconda3/bin/conda init bash
```

#cierra y abre el terminal, deberías encontrarte en:

(base) [UNIVERSIDADVIU\USUARIO ~]$

#(base): indica que nos encontramos en el entorno por defecto o base de Conda

---

Crear y navegar por los entornos

Aquí, los "entornos" representan la infraestructura en la que nuestro ordenador está operando actualmente. Esto incluye cosas como si existen ciertas variables y cómo se establecen (como nuestro PATH).

Entorno base

El entorno "base" de conda es, como suena, una especie de base de operaciones dentro de conda. No queremos instalar muchos programas complicados aquí, ya que mientras más cosas se agreguen, es más probable que algo termine teniendo un conflicto.

Crear un nuevo entorno

La forma más simple en que podemos crear un nuevo entorno conda es así:

```
conda create -n new-env
```

Donde el comando base es *conda create*, entonces estamos especificando el nombre de nuestro nuevo entorno con *-n* (aquí "**new-env**"). Comprobará algunas cosas y nos dirá dónde lo va a poner, cuando pulsemos "*y*" y enter, se creará.

Entrando en un entorno

Para entrar en ese entorno, tenemos que ejecutar

```
conda activate new-env
```

Y ahora podemos ver que nuestro prompt ha cambiado para tener **(new-env)** al principio, diciéndonos que estamos en ese entorno.

Si hubiéramos olvidado el nombre, o quisiéramos ver todos nuestros entornos, podemos hacerlo con:

```
conda env list
```

que imprimirá todos los entornos disponibles de conda, y tendrá un asterisco al lado del que estamos actualmente.

Salir de un entorno

Podemos salir de cualquier entorno conda en el que estemos actualmente ejecutando

```
conda deactivate
```

Creando un entorno con una versión específica de python

Por defecto, el comando conda create usará una versión de python. Pero podemos especificar una diferente en el comando si lo deseamos:

```
conda create -n python3 python=3.9
```

donde:

`conda create` - este es nuestro comando base

`-n python3` - estamos nombrando el entorno "python3"

`python=3.9` - aquí estamos especificando la versión de python a utilizar dentro del entorno

Podemos ver que ha funcionado comprobando nuestra versión de python dentro y fuera de estos entornos:

```
conda activate base
python --version
conda activate new-env
python --version
conda activate python3
python --version
```

Si estuviéramos tratando de usar un programa que sólo estuviera disponible en python 2, podríamos crear un entorno conda preparado para trabajar con él de esta manera sin estropear todas las cosas que usamos que dependen de python 3.

Es un poco difícil apreciar este tipo de cosas a menos que te hayas enfrentado a las consecuencias antes…

Eliminando un entorno

Podemos eliminar un entorno con:

```
conda deactivate # no podemos estar dentro del entorno que queremos eliminar
conda env remove -n python3
```

Buscando e instalando paquetes

Para instalar los paquetes (herramientas/programas) que queremos, usamos el comando conda install. Pero necesitamos decirle a conda donde buscarlo (conda almacena y busca los paquetes en diferentes "canales/repositorios"), y necesitamos saber cómo se llama la herramienta que buscamos en el repositorio de conda (normalmente será el nombre del programa con el que estamos familiarizados). Imaginemos que queremos instalar el programa de predicción de genes codificadores de proteínas prodigal.

Más sobre prodigal:

El articulo:

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-119

Repositorio en GitHub:

https://github.com/hyattpd/Prodigal

Cambiemos a nuestro "new-env" y comprobemos si prodigal ya está ahí:

```
conda activate new-env
```

```
prodigal -v
```

Y obtenemos un error de "comando no encontrado" si este no está instalado en nuestro sistema y accesible en nuestro entorno actual.

Veamos una forma común de encontrarlo e instalarlo a través de conda.

Búsqueda de paquetes

Lo más sencillo para saber si un programa es instalable por conda o no, es simplemente buscar en un navegador web "conda install" más el programa que estoy buscando. Por ejemplo, la búsqueda de "conda install prodigal" en google trae como resultado la página del paquete prodigal en anaconda.org (que es donde están alojados todos estos programas). Anaconda.org es también un gran lugar para buscar si tenemos problemas para encontrar un programa.

https://anaconda.org/

Instalación de paquetes

Mirando la página de paquetes prodigal, podemos ver que hay instrucciones para instalar a través de conda. Estas especifican el canal apropiado para buscar con *-c*. Así que podemos ejecutar esta línea para instalar prodigal:

```
conda install -c bioconda prodigal
```

A pesar de lo que los ejemplos de instalación muestran en las páginas de anaconda, como la de arriba, para que muchos paquetes de bioconda se instalen correctamente, es necesario establecer más canales en una jerarquía específica (documentación de bioconda aquí). Algunas instalaciones funcionan bien a pesar de todo (como es el caso de prodigal), pero otras fallarán, y aún peor, algunas tendrán éxito, pero tendrán problemas ocultos. Esto se debe a que, aunque prodigal puede estar en el canal de bioconda, algunas de sus dependencias pueden encontrarse en otros lugares o en múltiples lugares, y tener esos 3 canales establecidos en la prioridad adecuada ayuda a asegurar que se están utilizando las versiones correctas de todo. Es más fácil y seguro simplemente ejecutar el comando de instalación especificando todos los canales necesarios para ello en una sola línea así

```
conda install -c conda-forge -c bioconda -c defaults prodigal
```

De esta manera obtenemos lo que se necesita cada vez, y no tenemos que preocuparnos de cambiar la jerarquía de canales subyacente de conda en cada sistema en el que lo usamos, como demuestra la documentación de bioconda.

Esto muestra en pantalla, por ejemplo, qué y qué versiones de cosas se van a instalar, y dónde se van a instalar (que terminará con el nombre del entorno que hemos creado). Después de introducir y y pulsar return/enter, el programa se instala en este entorno, y podemos comprobar de nuevo prodigal, sólo que esta vez con éxito:

```
prodigal -v
```

Y ahora, como el programa está instalado en este entorno, obtenemos la información de la versión impresa como es debido. Si nos movemos fuera de este entorno, no se encontrará de nuevo (a menos que esté instalado allí también):

```
conda deactivate
```

```
prodigal -v
```

Es interesante destacar que la página de prodigal en github no lista la instalación de conda. Esto no es tan raro, ya que los desarrolladores no son siempre la gente que hace que sus programas sean instalables por conda. No ver las instrucciones de instalación de conda en la página de instalación de un programa en particular, no significa que no esté disponible a través de conda.

Si quisiéramos ver los paquetes del ambiente en el que nos encontramos:

```
conda list
```

Desinstalando un paquete

Si quisiéramos eliminar prodigal, tenemos primero que estar dentro del ambiente que lo tiene, así que cambiemos a nuestro nuevo ambiente:

```
conda activate new-env
```

Y luego ejecutaríamos

```
conda uninstall prodigal
```

Una vez que confirmamos con "*y*" y pulsamos return, prodigal vuelve a desaparecer:

```
prodigal -v
```

Y los paquetes del ambiente ya no nos encontramos:

```
conda list
```

Instalar una versión específica de un paquete

Si quisiéramos especificar una versión diferente de prodigal para instalar, primero necesitamos saber que existe en conda. Podemos ver todos los tipos de versiones disponibles así:

```
conda search -c bioconda prodigal
```

Esto lista 2 versiones diferentes disponibles (2.6.2 y 2.6.3, con 3 diferentes "builds" para cada una. Un build aquí es cuando el programa no ha cambiado, pero la forma en que fue empaquetado para conda ha cambiado. Si quisiéramos instalar la v2.6.2 en lugar de la v2.6.3 como obtuvimos arriba, podríamos especificarlo así

```
conda install -c conda-forge -c bioconda -c defaults prodigal=2.6.2
```

Al igual que arriba cuando especificamos la versión de python que queríamos, podemos colocar un signo de igualdad y luego la versión que queramos de cualquier programa que estemos instalando.

Si confirmamos esto introduciendo "*y*" y pulsando "return" finaliza la instalación, y podemos ver que ahora tenemos esta versión más antigua:

```
prodigal -v
```

Canales de instalación en conda

La jerarquía de canales en Conda determina cómo se buscan y seleccionan los paquetes cuando se instalan o actualizan. Brevemente:

- *Ubicación de los Paquetes*: Los canales son ubicaciones donde se almacenan los paquetes. Son URLs a directorios que contienen paquetes de Conda.
- *Prioridad de los Canales*: Conda busca paquetes en un conjunto predeterminado de canales, y por defecto, da preferencia a los paquetes de un canal con mayor prioridad sobre los de canales de menor prioridad (conda-forge > bioconda > defaults).
- *Resolución de Conflictos*: Si diferentes canales tienen el mismo paquete, Conda resuelve los conflictos de canales, usando el canal predeterminado o no usando paquetes en común.
- *Orden de Instalación*: Conda recopila todos los paquetes con el mismo nombre en todos los canales listados y los ordena por prioridad de canal, número de versión y número de compilación. Instala el primer paquete de la lista que cumple con las especificaciones de instalación.

Cuando hacemos algo en conda, automáticamente busca en nuestros canales almacenados, en un orden jerárquico específico: conda-forge > bioconda > defaults. Tuvimos que especificar *-c bioconda* en el comando anterior cuando instalamos prodigal porque ese canal aún no estaba en nuestros canales almacenados. Podemos establecer esto de antemano, o podemos especificarlo cuando instalamos algo.

La mayoría de las *cosas* que los biólogos querremos usar se encuentran en el canal bioconda. Las instrucciones de la página del paquete prodigal en las instrucciones de anaconda.org no son en realidad 100% ideales. Mirando la documentación de bioconda podemos ver que debemos especificar la prioridad de nuestro canal de forma que se busque en *conda-forge* antes que en *bioconda*, que se busca antes que en *defaults*.

Aunque funcionó bien con prodigal cuando sólo le dimos el canal *-c bioconda*, otros programas podrían tener problemas. Así que es mejor seguir la documentación de bioconda y especificar todos los canales en el orden correcto. Aquí hay dos maneras de hacerlo.

Podemos especificar estos canales en el comando de instalación así

```
conda activate new-env # asegurándonos de que estamos en nuestro nuevo entorno
conda install -c conda-forge -c bioconda -c defaults prodigal
```

(Aunque en este caso nos va a preguntar si queremos actualizar la versión que tenemos a una más nueva, podemos darle a "*n*" y return).

O podemos configurar los canales con antelación como demuestra la documentación de bioconda (cuando se hace así, el último que añadimos tiene la mayor prioridad):

```
conda config --get channels                 #para comparar con el posterior resultado
conda config --add channels defaults        # no hay que preocuparse por la advertencia
conda config --add channels bioconda
conda config --add channels conda-forge
```

Y podemos ver nuestra lista de canales y el orden de prioridad con lo siguiente

```
conda config --get channels
```

Ahora podríamos instalar desde bioconda sin necesidad de especificar los canales si queremos, por ejemplo

```
conda install prodigal
```

(De nuevo esto nos preguntará si queremos actualizar. Si tuviéramos la misma versión que la última, simplemente diría que ya está instalada. Podemos pulsar "*n*" y enter).

Volvamos a nuestro entorno base antes de seguir adelante:

```
conda deactivate
```

Exportar y crear entorno mediante archivos *.yaml*

En los ejemplos anteriores, creamos un nuevo entorno y proporcionamos los programas/versiones que queríamos mediante los distintos comandos.

Conda permite exportar los ambientes creados a archivos *.yaml* y crear ambientes desde estos archivos.

Esto significa que yo puedo crear un ambiente y exportarlo para compartirlo. Esto es fantástico para la reproducibilidad de flujos de trabajo. Puedo crear un ambiente para mi proyecto, instalar todo lo necesario para ejecutar el pipeline que deseo y posteriormente exportar y compartir ese ambiente de conda y el pipeline, permitiendo su reproducibilidad.

💡 Consejo: Recuerda exportar periódicamente tus ambientes favoritos a una nube de almacenamiento para que, si hay problema con el equipo, puedas tenerlo funcionando rápidamente utilizando la función de importación de ambientes de conda.

A veces es conveniente usar un archivo yaml que contenga toda la información sobre los programas/versiones que queremos, y podemos simplemente darle ese archivo al comando *conda create*.

Aquí hay un ejemplo de un archivo yaml formateado por conda:

```
name: prodigal_env

channels:

  - conda-forge
```

```
  - bioconda
  - defaults
dependencies:
  - prodigal=2.6.2
prefix: /home/jtronchoni2/miniconda3/envs/new-env
```

Aquí, estamos especificando el "nombre", que se convierte en el nombre del entorno de conda (es decir, lo que le daremos al comando *conda activate* ...), los canales (deben tener los 3 especificados en ese orden), y luego las "dependencias" son los programas y/o versiones que queremos.

Para crear un entorno a partir de un archivo yaml:

```
conda env create -f prodigal_env.yml
```

Donde:

`conda env create` - nuestro comando base, cuando proporcionamos un archivo yaml, necesitamos añadir la parte 'env' aquí, a diferencia de lo anterior donde sólo usamos conda create.

`-f` - aquí es donde proporcionamos el archivo que contiene la información sobre el entorno que queremos crear

Una vez hecho esto, podemos activarlo de la misma manera

```
conda activate prodigal_env
```

Comprobamos que contiene prodigal con:

```
prodigal –v
```

💡 Consejo: El lugar adecuado donde revisar la documentación sobre comandos y parámetros es siempre el manual oficial de la herramienta que estamos utilizando y por tanto el recomendable para salir de dudas. A pesar de ello, las IAs resultan muy útiles para hacernos una idea rápida cuando tengamos una duda. Prueba a ejecutar este prompt en tu IA favorita:

Explícame que significan estos comandos y parámetros en la siguiente línea de código:

conda env create -f prodigal_env.yml

Si lo que queremos es exportar nuestro entorno. Activamos el entorno que queremos exportar y entonces:

```
conda env export --from-history>environment.yml
```

Donde environment.yml será el nombre del archivo de exportación. Aconsejable cambiarlo por el nombre del environment que estás exportando.

Como realmente no lo necesitamos, recuerda que podemos eliminar un entorno como el que se muestra a continuación, sólo tenemos que desactivarlo primero:

```
conda deactivate
```

```
conda env remove -n prodigal_env
```

ATENCIÓN:

En algunos casos podéis encontrar que el prompt no se muestra de forma habitual al iniciar el terminal con conda. En ocasiones conda interfiere con el archivo de iniciación del bash de nuestro ordenador. Para solucionarlo:

```
conda init bash
```

Otros recursos de Conda

Documentación principal de Conda

Guía principal del usuario de Conda

Tutorial de iniciación de la gente de Conda

Comandos principales de Conda

Anaconda.org