Explicación 0: Instalación de Ruby

Taller de Tecnologías de Producción de Software - Ruby

Cursada 2020

En esta explicación se trata de explicar en una guía paso a paso cómo preparar un ambiente de desarrollo local para Ruby.

Gestores de ambiente

Los gestores de ambiente, también llamados *manejadores de versiones*, facilitan el uso de múltiples versiones de un lenguaje (Ruby, en nuestro caso) en un mismo sistema.

Idealmente, el gestor de ambiente será una herramienta que no se note pero que estará presente en la shell, cambiando la versión de Ruby automáticamente (acorde a reglas bien definidas) o según las indicaciones particulares del usuario.

Los más populares:

- RVM
  - Ruby enVironment Manager.
  - Herramienta pionera en su función.
- rbenv
  - Ruby environment.
  - Alternativa más liviana y con menores pre-requisitos que RVM.
  - Extensible mediante plugins.

rbenv

En esta explicación nos vamos a centrar en **rbenv**, el gestor que desde la cátedra recomendamos utilizar.

Instalación de rbenv

Para instalar este gestor, se debe contar con el comando `git`, las librerías necesarias para compilar programas escritos en C y algunas dependencias más. En la versión actual de Ubuntu (20.04 al momento de escribir esta guía), por ejemplo, los paquetes necesarios pueden instalarse con los siguientes comandos:

```
# apt update -qq
# apt install -y git \
                 build-essential \
                 autoconf \
                 bison \
                 curl \
                 lib{ssl,yaml,sqlite3}-dev \
                 libreadline{8,-dev} \
                 zlib1g{,-dev}
```

Una vez que se cuente con los requisitos, se puede proceder a realizar la instalación.

El primer paso es clonar el repositorio de rbenv localmente. Esto se hará en el directorio `.rbenv` dentro del directorio personal (`$HOME`) del usuario que se deseee:

```
$ git clone https://github.com/sstephenson/rbenv.git ~/.rbenv
```

Acto seguido, se procederá a situarse en ese directorio y a compilar una extensión que acelerará el uso diario de rbenv (este paso es opcional):

```
$ cd ~/.rbenv
$ src/configure && make -C src
```

Después de esto, se debe configurar la shell para que cargue rbenv cada vez que se inicie, dejando disponibles los *shims* que rbenv genera para poder cambiar automáticamente las versiones de las herramientas. Esto se hace agregando al archivo de perfil (`~/.bashrc` en este caso) de la shell 2 líneas:

- una que agrega al principio de la variable de ambiente `$PATH` la ruta al directorio `bin` de rbenv (donde están los shims ejecutables), y
- otra que carga el script de inicialización de rbenv.

Esto puede hacerse ejecutando los siguientes comandos:

```
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
```

> **Nota:** el archivo de perfil del usuario puede variar según el Sistema Operativo que se utilice y según el intérprete de comandos (Bash, Zsh, Ksh, entre otros). El archivo al que se hace referencia aquí es a uno que se cargue cuando la shell se inicie. En otras distribuciones de GNU/Linux, por ejemplo, el archivo es `~/.bash_profile`.

Y por último, se debe reiniciar la shell para que tome los cambios. Esto puede hacerse saliendo (con

el comando `exit` o presionando Ctrld, por ejemplo) de la sesión de shell y volviendo a entrar, o bien ejecutando el siguiente comando:

```
$ exec $SHELL -l
```

Luego de esto, se puede comprobar si la instalación de rbenv fue exitosa al ejecutar el siguiente comando:

```
$ type rbenv
```

La salida esperada para ese comando debe ser similar a esto:

```
rbenv is a function
rbenv ()
{
    local command;
    command="${1:-}";
    if [ "$#" -gt 0 ]; then
        shift;
    fi;
    case "$command" in
        rehash | shell)
            eval "$(rbenv "sh-$command" "$@")"
        ;;
        *)
            command rbenv "$command" "$@"
        ;;
    esac
}
```

Si esa fue la salida, se pueden verificar las versiones de Ruby disponibles con el siguiente comando:

```
$ rbenv versions
```

Normalmente, esto debería indicar que no se encuentran versiones de Ruby en el sistema:

```
Warning: no Ruby detected on the system
```

Para poder instalar versiones de Ruby, se debe continuar con el siguiente apartado.

Instalación de ruby-build

La herramienta rbenv por si sola nos permite cambiar de versiones de Ruby, pero no instala nuevas versiones. Para poder hacer eso, se debe contar con el *plugin* `ruby-build` para rbenv.

Su instalación es muy sencilla:

```
$ git clone https://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins
    /ruby-build
```

A partir de esto se pueden ver las versiones de Ruby disponibles para instalar ejecutando el siguiente comando:

```
$ rbenv install -l
```

Y si se quiere instalar cualquiera de las versiones que se listan, sólo basta especificar la versión y `ruby-build` se encargará de instalarla. Por ejemplo, para instalar la versión `2.7.1` se debe ejecutar este comando (que puede tardar):

```
$ rbenv install 2.7.1
```

Si todo salió bien, se puede corroborar que se disponga de la versión deseada con este comando:

```
$ rbenv versions
```

La salida del comando debería incluir `2.7.1`. De ser así, la instalación fue exitosa.

Aún cuando se tenga instalada una versión de Ruby, si se intenta ejecutar su intérprete es probable que se obtenga un error como el siguiente:

```
$ ruby -v
rbenv: ruby: command not found

The `ruby' command exists in these Ruby versions:
  2.7.1
```

Esto se debe a que aún no se ha especificado qué versión de Ruby utilizar. Esto se hará en la próxima sección.

> **Nota:** esta forma de instalar rbenv hace deseable la actualización periódica de rbenv y ruby-build para poder tener acceso a las últimas versiones del lenguaje.
>
> Para esto, basta con ejecutar cada tanto los siguientes comandos con el usuario que se haya instalado rbenv:
>
> ```
> $ cd ~/.rbenv
> $ git pull origin master
> $ cd plugins/ruby-build
> $ git pull origin master
> ```

Uso

El uso de este gestor es, en general, totalmente transparente al usuario y automático. La herramienta se encarga de cambiar la versión de Ruby a utilizar acorde a reglas bien definidas, que se consideran en el siguiente orden (toma la la primera versión que se especifique respetando este orden de prioridad):

1. El valor de la variable de ambiente `$RBENV_VERSION`. Esta se llama versión de `shell`.
2. El contenido del primer archivo llamado `.ruby-version` encontrado desde el directorio donde reside el script Ruby que se esté ejecutando, o subiendo de a un nivel en los directorios padre hasta alcanzar la raiz del *filesystem* (el directorio `/`).
3. El contenido del primer archivo llamado `.ruby-version` encontrado desde el directorio actual donde esté posicionado el usuario, o subiendo de a un nivel en los directorios padre hasta alcanzar la raiz del *filesystem*. Esta se llama versión `local`.
4. La versión `global` especificada en el archivo `~/.rbenv/global`.

Rbenv provee algunos comandos que permiten generar o modificar la versión de Ruby deseada en los distintos contextos:

- La versión `shell` se puede especificar con el comando `rbenv shell 2.7.1`.
- La versión `local` se puede especificar con el comando `rbenv local 2.7.1`.
- La versión `global` se puede especificar con el comando `rbenv global 2.7.1`.

Con esto en mente, es conveniente definir la versión `global` de Ruby que se desea utilizar:

```
$ rbenv global 2.7.1
```

Luego de esto, se puede corroborar que ahora sí se puede utilizar el intérprete de Ruby en la versión deseada:

```
$ ruby -v
```

Ese comando debería tener una salida similar a la siguiente:

```
ruby 2.7.1p83 (2020-03-31 revision a0c7c23c9c) [x86_64-linux]
```

Con esto se concluye con la instalación del ambiente de Ruby. En cualquier momento, se pueden instalar otras versiones del intérprete de Ruby ejecutando simplemente:

```
$ rbenv install VERSION
```

Donde `VERSION` es la versión que se desea instalar, y que debe estar disponible entre las versiones que aparecen al ejecutar `rbenv install -l`.

Paso final: prueba

Una vez que se tiene el ambiente de desarrollo en Ruby instalado y configurado, se lo puede probar ejecutando un script escrito en Ruby. Para ello, se puede tomar el que se incluye en el repositorio de explicaciones prácticas o escribir uno.

Para ejecutar cualquier archivo utilizando el intérprete de Ruby, se usa un comando como el siguiente:

```
$ ruby ruta/al/archivo.rb
```

Por ejemplo, para ejecutar el script `explicaciones/00-ambiente/hello.rb` del repositorio de explicaciones prácticas:

```
$ ruby explicaciones/00-ambiente/hello.rb
```

Notar que la extensión `.rb` es simplemente una convención, los archivos Ruby son archivos de texto plano, más allá de su extensión (o la ausencia de la misma); es decir que el script podría llamarse `hello`, `hello.ruby` o de cualquier otra manera, y sin importar eso si el archivo es de texto plano y el contenido es código Ruby válido, el intérprete lo ejecutará sin problemas.