# System Administration

1. What is the difference between yum update and yum upgrade? (3 %)

- yum upgrade 會更新已安裝的套件，並且連同一些過舊即將洮汰的套件也一起更新。 yum update 則是會更新已安裝的套件，但不一定會連同一些過舊即將洮汰的套件也一起更新。

2. Are there any flags you can add to yum update to make it the same as yum upgrade? (3%)

- 對 yum update 加上 --obsoletes 這個參數後就會連同一些過舊即將洮汰的套件也一起更新了。

3. What is the difference between yum remove and yum autoremove? (3%)

- yum remove <package name> 會移除符合後面參數所提供的套件名稱的套件。 yum autoremove [package name] 則是，如果有提供套件名，則會移除符合後面參數所提供的套件名稱的套件，並移除沒有用到的相依套件；否則移除系統中遺留下來沒有用的套件。

4. How to prevent yum autoremove from removing a specific package? (3%)

- yumdb set reason user <specific package name>，之後 yum 就會把那個特定的套件視為是你自己安裝的，然後 yum autoremve 的時候就不會把他給移除了。

5. How to search for a package, say vim? (3%)

- yum search <package name>

6. Sometimes the program name differs from the package name. For example, program nfsstat is packaged in the nfs-utils RPM. How to perform such search, given only the program name? (3%)

- yum provides <package name>

7. How to recursively list all dependencies of a package? (3%)

- repoquery --tree-requires --recursive <package name>

8. Aside from system-wide package manager like yum, many programming language communities have their own package managers. Rust has Cargo; Python has Pip; Node.js has NPM, etc. Both system-wide and language-specific package managers produce modules for the language in concern. For example, Sphinx, a popular documentation generator written in Python, can be installed in CentOS 7 by either sudo yum install python-sphinx or sudo pip install sphinx. If you want to install a Python module that are provided by both CentOS 7 repository and PyPI, what are the pros and cons of each method? (8%)

- 如果使用 CentOS 7 repository 本身所提供的套件，在正常情況下他都會比較穩定且安全，且能正常 Work 的，因為這個套件是有經由 CentOS 公司確認過沒有問題才放上來的。但是 PyPI 的話則是一個大家都可以自行傳套件上去的地方，因此必較不安全且不穩定。
- 不過如果使用 CentOS 7 repository 本身所提供的套件一般都只能裝在整個系統上，不能設定虛擬環境 (Virtualenv)。而 PyPI 的好處就是可以針對每個專案都使用不同的套件，且專案間的套件不會互相影響。

9. What's wrong with the following sequence of commands, assuming that pip has been installed from

EPEL repository? Please show any warnings issued by pip or yum and explain the reason. (8%)

```
sudo yum install -y python-jinja2
... some weeks later ....
sudo pip list | grep jinja         # 1
sudo pip uninstall -y jinja2       # 2
... some weeks later ...
sudo yum remove -y python-jinja2   # 3
```

- 1. jinja 打錯了，應該要輸入 Jinja2
  2. 不應該用 pip 把用 yum 安裝的套件移除掉，因為這樣 yum 不會記錄到這個套件已經移除了，之後處理套件相依性時可能會有問題。
  3. 因為之前用 pip 把 jinja2 給移除掉了，所以這時候在移除 python-jinja2 時會噴警告。並且應該使用 autoremove 來把用不到的套件移除。

10. Some Ruby Gems build “native extensions”, C codes embedded in Ruby, when installed by gem install. The build process includes generating a Makefile and uses that to invoke C or C++ compiler. But what if you are on a minimal CentOS without make, gcc, gcc-c++ being installed? Or a library is missing? Another issue of using gem is that some Gems expect the Ruby version different from the one provided by the ruby RPM. Compare installing Wikicloth via yum (in the EPEL repository) and gem. Describe what happens when you are on a minimal CentOS 7 without aforementioned build tools or on a hardware device with poor CPU. Describe version conflicts you meet, if any. Finally, list the installation destination of each package manager and compare with that of Python in the previous problem (Hint: rpm -ql). (8%)

- 首先我們試著用 yum 去安裝 wikicloth

```
# yum install rubygem-wikicloth
```

- 因為 yum 是直接把 binary 載下來給你使用，因此並不會需要在本機上編譯，所以就能成功地安裝起來。
- 接者我們試著用 gem 去安裝 wikicloth

```
# yum install -y ruby
# gem install wikicloth
Fetching: builder-3.2.3.gem (100%)
Successfully installed builder-3.2.3
Fetching: expression_parser-0.9.0.gem (100%)
Successfully installed expression_parser-0.9.0
Fetching: unf_ext-0.0.7.4.gem (100%)
Building native extensions.  This could take a while...
ERROR:  Error installing wikicloth:
        ERROR: Failed to build gem native extension.

    /usr/bin/ruby extconf.rb
mkmf.rb can't find header files for ruby at /usr/share/include/ruby.h
```

```
Gem files will remain installed in /usr/local/share/gems/gems/unf_ext-0.0.7.4
for inspection.
Results logged to /usr/local/share/gems/gems/unf_ext-
0.0.7.4/ext/unf_ext/gem_make.out
```

- 從上面我們會發現因為我們忘了裝ruby-devel

```
# yum install -y ruby-devel
[joe@localhost ~]$ sudo gem install wikicloth
Building native extensions.  This could take a while...
ERROR:  Error installing wikicloth:
        ERROR: Failed to build gem native extension.

    /usr/bin/ruby extconf.rb
checking for main() in -lstdc++... *** extconf.rb failed ***
Could not create Makefile due to some reason, probably lack of necessary
libraries and/or headers.  Check the mkmf.log file for more details.  You may
need configuration options.

Provided configuration options:
        --with-opt-dir
        --without-opt-dir
        --with-opt-include
        --without-opt-include=${opt-dir}/include
        --with-opt-lib
        --without-opt-lib=${opt-dir}/lib64
        --with-make-prog
        --without-make-prog
        --srcdir=.
        --curdir
        --ruby=/usr/bin/ruby
        --with-static-libstdc++
        --without-static-libstdc++
        --with-stdc++lib
        --without-stdc++lib
/usr/share/ruby/mkmf.rb:434:in `try_do': The compiler failed to generate an
executable file. (RuntimeError)
You have to install development tools first.
        from /usr/share/ruby/mkmf.rb:519:in `try_link0'
        from /usr/share/ruby/mkmf.rb:534:in `try_link'
        from /usr/share/ruby/mkmf.rb:714:in `try_func'
        from /usr/share/ruby/mkmf.rb:944:in `block in have_library'
        from /usr/share/ruby/mkmf.rb:889:in `block in checking_for'
        from /usr/share/ruby/mkmf.rb:340:in `block (2 levels) in postpone'
        from /usr/share/ruby/mkmf.rb:310:in `open'
        from /usr/share/ruby/mkmf.rb:340:in `block in postpone'
        from /usr/share/ruby/mkmf.rb:310:in `open'
        from /usr/share/ruby/mkmf.rb:336:in `postpone'
```

```
        from /usr/share/ruby/mkmf.rb:888:in `checking_for'
        from /usr/share/ruby/mkmf.rb:939:in `have_library'
        from extconf.rb:6:in `<main>'


Gem files will remain installed in /usr/local/share/gems/gems/unf_ext-0.0.7.4
for inspection.
Results logged to /usr/local/share/gems/gems/unf_ext-
0.0.7.4/ext/unf_ext/gem_make.out
```

- 然後我們就會發現他因為沒有 g++ 因此裝不了。因此我們改用 yum 安裝 wikicloth
- 然後為了要讓 gem 可以正常的裝 wikicloth，因此在電腦上裝 g++。

```
# yum install -y gcc gcc-c++
# gem install wikicloth
Building native extensions.  This could take a while...
Successfully installed unf_ext-0.0.7.4
Fetching: unf-0.1.4.gem (100%)
Successfully installed unf-0.1.4
Fetching: twitter-text-1.14.5.gem (100%)
Successfully installed twitter-text-1.14.5
Fetching: mini_portile2-2.1.0.gem (100%)
Successfully installed mini_portile2-2.1.0
Fetching: nokogiri-1.7.2.gem (100%)
ERROR:  Error installing wikicloth:
        nokogiri requires Ruby version >= 2.1.0.
```

- 結果我們發現其中 nokogiri 這個套件所要求的 ruby version 要 >= 2.1.0，因此我們就安裝舊版的 nokogiri來解決這個問題。

```
# gem install nokogiri:1.6.1
Fetching: nokogiri-1.6.1.gem (100%)
Building native extensions.  This could take a while...
Successfully installed nokogiri-1.6.1
Parsing documentation for nokogiri-1.6.1
Installing ri documentation for nokogiri-1.6.1
# gem install wikicloth
Fetching: htmlentities-4.3.4.gem (100%)
Successfully installed htmlentities-4.3.4
Fetching: wikicloth-0.8.3.gem (100%)
Successfully installed wikicloth-0.8.3
Parsing documentation for htmlentities-4.3.4
Installing ri documentation for htmlentities-4.3.4
Parsing documentation for wikicloth-0.8.3
Installing ri documentation for wikicloth-0.8.3
2 gems installed
```

- 然後我們就發現 wikicloth 成功的用 gem 裝起來了。
- 然後這個是 yum 安裝 wikicloth 的位置

```
# rpm -ql rubygem-wikicloth
...
/usr/share/gems/gems/wikicloth-0.8.0/lib/wikicloth.rb
...
```

- 這個是 gem 安裝 wikicloth 的位置

```
# gem which wikicloth
/usr/local/share/gems/gems/wikicloth-0.8.3/lib/wikicloth.rb
```

- 這個是 gem 使用一般使用者權限安裝 wikicloth 的位置

```
$ gem which wikicloth
/home/joe/.gem/ruby/gems/wikicloth-0.8.3/lib/wikicloth.rb
```

- 然後這個是用 yum 安裝的 jinja2 的位置

```
# rpm -ql  python-jinja2
...
/usr/lib/python2.7/site-packages/jinja2
...
```

- 這個是用 pip 查詢 jinja2 的位置(沒有把 yum 裝的 jinja2 移除)

```
$ pip show Jinja2
...
Location: /usr/lib/python2.7/site-packages
...
```

- 這個是用 pip 安裝 jinja2 的位置(安裝前把 yum 裝的 jinja2 先移除)

```
$ pip show Jinja2
...
Location: /usr/lib64/python2.7/site-packages
...
```