dnf module

Table of Contents

Manage modules with dnf.

The DNF modularity system provides more ways of building and consuming software. The build side adds more binaries, packages and metadata to the DNF system. For consumers, the DNF modularity system solves the problem of how to maintain older versions of a product. More information is in the fedora page, system manual (`man 7 dnf.modularity`), and RHEL 8 module chapter.

A traditional approach to supporting software is to support the latest version only. The DNF system works well with this approach, installing the latest version of a package. For Fedora, an OS for users at the leading edge of open source software, a DNF update keeps the system up-to-date. For Red Hat, an OS for the enterprise, requirements for security, reliability and scale may not fit with leading-edge products. A DNF update of older software may not welcome.

The DNF modularity system solves the problem of how to maintain older versions of a product. An upgrade to the latest version is not always welcome in an enterprise environment, because a stable application, that works reliably for many years, can cause fewer maintenance headaches than the latest application. Software is kept up-to-date with security patches and bug fixes, but upgrades that change features may break applications. Slow and boring beats failing fast.

DNF also has a plug-in system. DNF plug-ins and DNF modules are not related.

a DNF module

The DNF modularity system treats a complex collection of software as one discrete component. The Postgresql database server for RHEL 8 is version 10. Postgresql is available as a DNF module. The module provides three streams, for the older version 9, the default version 10 and Postgresql's current version 12. Only one of these three streams should be installed. If more than one Postgresql version is required, use containers to keep the streams apart.

Many DNF commands like `dnf install`, `dnf provides` and `dnf info` work for modules by changing *dnf* to *dnf module*. Run `dnf module --help` for a list of commands and options.

The appstream repo contains dozens of modules.

```
[nick@guest1 ~]$ dnf module list
...
Name                Stream      Profiles   Summary
389-ds              1.4                    389 Directory Server (base)
ant                 1.10 [d]    common [d] Java build tool
container-tools     rhel8 [d]   common [d] Common tools and dependencies for
container ru
                                           ntimes
...
varnish             6 [d]       common [d] Varnish HTTP cache
virt                rhel [d][e] common [d] Virtualization module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
[nick@guest1 ~]$
```

List only postgresql modules.

```
[nick@guest1 ~]$ sudo dnf module list postgresql
Updating Subscription Management repositories.
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)   39 kB/s | 2.8 kB     00:00
Red Hat Ansible Engine 2 for RHEL 8 x86_64 (RPMs)          31 kB/s | 2.4 kB     00:00
Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)      33 kB/s | 2.4 kB     00:00
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name            Stream      Profiles               Summary
postgresql      9.6         client, server [d]     PostgreSQL server and client module
postgresql      10 [d]      client, server [d]     PostgreSQL server and client module
postgresql      12          client, server [d]     PostgreSQL server and client module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
[nick@guest1 ~]$
```

One stream and one profile are labelled **[d]** (default). If the user enters the command `dnf module install postgresql`, the **default** stream is the one that will be enabled and installed. It's the same for profiles - the *server* packages will be installed, not the *client* packages.

module stream

A dnf module stream is a collection of RPM packages for one application. If a module has more than one version, each version gets its own stream. The 389-ds module only covers version 1.4 and has one stream. The nodejs module has two streams, one for major version 10 and one for v12. Nginx has different streams for minor versions - 1.14 and 1.16.

A module can have many streams, but only one can be enabled. Enabling a stream is a way of

telling DNF to which one to install, and which one to keep maintained by installing package updates. If the admin installs a module without picking a stream, the default stream is enabled and installed.

Enable Postgresql stream 12 (version 12).

The module identifier used here is in the form NAME:STREAM. A module identifier can have a lot of fields (NAME:STREAM:VERSION:CONTEXT:ARCH/PROFILE, or NSVCA naming convention), but most are never seen by users.

```
[nick@guest1 ~]$ sudo dnf module enable postgresql:12
...
================================================================================
====
 Package              Architecture        Version              Repository
Size
================================================================================
====
Enabling module streams:
 postgresql                               12

Transaction Summary
================================================================================
====

Is this ok [y/N]: y
Complete!
[nick@guest1 ~]$
[nick@guest1 ~]$ sudo dnf module list postgresql
...
Name            Stream      Profiles                Summary
postgresql      9.6         client, server [d]      PostgreSQL server and client module
postgresql      10 [d]      client, server [d]      PostgreSQL server and client module
postgresql      12 [e]      client, server [d]      PostgreSQL server and client module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
[nick@guest1 ~]$
```

module profile

Every module has an installation profile. A profile is a collection of packages for one use case. Most modules have only one profile called **common**. Postgresql has two profiles, **client** and **server**. The client profile installs one package, *postgesql*, which pulls in a few dependencies . The server profile is similar, installing *postgesql-server*.

```
[nick@guest1 ~]$ dnf module info postgresql --profile
...
Name   : postgresql:10:820190104140132:9edba152:x86_64
client : postgresql
server : postgresql-server

Name   : postgresql:12:8010120191120141335:e4e244f9:x86_64
client : postgresql
server : postgresql-server

Name   : postgresql:9.6:820190104140337:9edba152:x86_64
client : postgresql
server : postgresql-server
[nick@guest1 ~]$
```

A profile is a collection of packages for one purpose, so it can't be enabled or disabled.

install a module profile

Install Postgresql, stream 12, profile client.

Stream 12 is enabled, so that does not need to be included. These commands are equivalent.

- dnf module install postgresql/client
- dnf module install postgresql:12/client
- dnf install @postgresql:12/client

The '*' wildcard character works, so you can also install both profiles with `sudo dnf module install postgresql/*`

```
[nick@guest1 ~]$ sudo dnf module install postgresql/client
...
================================================================================
====
 Package
        Arch   Version                              Repository
Size
================================================================================
====
Installing group/module packages:
 postgresql
        x86_64 12.1-2.module+el8.1.1+4794+c82b6e09 rhel-8-for-x86_64-appstream-rpms
1.4 M
Installing dependencies:
 libpq  x86_64 12.1-3.el8                          rhel-8-for-x86_64-appstream-rpms
195 k
Installing module profiles:
 postgresql/client


Transaction Summary
================================================================================
====
Install  2 Packages
...
Complete!
[nick@guest1 ~]$
[nick@guest1 ~]$ sudo dnf module list postgresql
...
Name           Stream     Profiles                 Summary
postgresql     9.6        client, server [d]       PostgreSQL server and client
module
postgresql     10 [d]     client, server [d]       PostgreSQL server and client
module
postgresql     12 [e]     client [i], server [d]   PostgreSQL server and client
module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
[nick@guest1 ~]$
```

change streams

Switching to another stream is more complicated than running one command like `dnf module enable postgresql:9.6` or `dnf module install postgresql:9.6`. Follow this procedure.

- Remove module packages.
- Reset the module.
- Enable another stream.

- Install the profile.

These examples change streams from Postgresql stream 12 to stream 9.6.

remove module packages

Remove all the module packages.

```
[nick@guest1 ~]$ sudo dnf module remove --all postgresql
Updating Subscription Management repositories.
Dependencies resolved.
================================================================================
====
 Package
       Arch   Version                              Repository
Size
================================================================================
====
Removing:
 postgresql
       x86_64 12.1-2.module+el8.1.1+4794+c82b6e09 @rhel-8-for-x86_64-appstream-rpms
5.7 M
Removing unused dependencies:
 libpq x86_64 12.1-3.el8                          @rhel-8-for-x86_64-appstream-rpms
808 k
Disabling module profiles:
 postgresql/client

...
Complete!
[nick@guest1 ~]$
```

reset the module

Erase your configuration change.

```
[nick@guest1 ~]$ yum module reset postgresql
...
================================================================================
====
 Package              Architecture        Version              Repository
Size
================================================================================
====
Resetting modules:
 postgresql

Transaction Summary
================================================================================
====

Is this ok [y/N]: y
Complete!
[nick@guest1 ~]$
```

Check your work.

The `dnf distro-sync` command aligns all packages with repositories. It's a quick way of looking for anything unusual. When it finishes with *Nothing to do*, like this example does, packages are good.

```
[nick@guest1 ~]$ dnf module list postgresql
...
Name            Stream      Profiles                Summary
postgresql      9.6         client, server [d]      PostgreSQL server and client module
postgresql      10 [d]      client, server [d]      PostgreSQL server and client module
postgresql      12          client, server [d]      PostgreSQL server and client module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
[nick@guest1 ~]$
[nick@guest1 ~]$ sudo dnf distro-sync
...
Nothing to do.
Complete!
[nick@guest1 ~]$
```

enable another stream

Run `sudo dnf enable postgresql:9.6`.

install the profile

Run `dnf module install postgresql/client`.

maintenance

No special module treatment is needed for system operation. Running `dnf update` upgrades module packages to the latest versions provided by their streams.