Tools for Making Modules in Fedora

Flock Budapest, August 2019

Presented by:
Merlin Mathesius

Who is this talk for?

- Fedora package maintainers
- … who are familiar with RPM packaging tools
- … who are interested in module creation/building tools

What this talk is about

- A brief introduction to Fedora modules and associated terminology
- Some ideas about when you would want to create a module
- A few highlighted tools Fedora has to assist in creating and building modules

What this talk is about

- A brief introduction to Fedora modules and associated terminology
- Some ideas about when you would want to create a module
- A few highlighted tools Fedora has to assist in creating and building modules

What this talk is *not* about

- The pros and cons of the current Fedora Modularity design
- The edge cases when you should NOT create a module

What are modules?

Collections of packages.

With “special sauce!” (The module metadata.)

Managed on your system via **`dnf module`** … commands.

Terminology

Module A group of packages representing an application, language runtime, or other logical collection. (eg., “`nodejs`”, “`mysql`”, “`perl`”, etc.)

Stream A particular version of a module’s content. (eg., “`6`”, “`8`”, “`devel`”, “`latest`”, etc.)

Version The version of the module stream. Typically filled in by the module buildsystem in the form of concatenated release and timestamp. (eg., “`3120190807143546`”) Not to be confused with package versions or the stream name.

Terminology, continued

Profile Lists of packages representing a specific use case of the module stream. Can be installed as a set by referring to the profile name. (eg., “`client`”, “`server`”, “`minimal`”, etc.)

Defaults Each module can have a default stream that will be used if a specific stream is not explicitly requested. Similarly, each stream can have a default installation profile that is used if a specific profile is not requested.

When to use modules

- Solve “too fast/too slow” problem.
- Parallel availability of multiple versions of a package or bundle of packages.
- NOT parallel installability.

Discovering existing modules and their contents

Package level: same as always

- `dnf repoquery …`
- `dnf list …`
- `dnf search …`

Module level

- `dnf module list …`

Discovering existing modules/content, continued

Query MBS API

https://mbs.fedoraproject.org/module-build-service/1/module-builds/?name=***module***&stream=***stream***&short=true

Query PDC API

https://pdc.fedoraproject.org/rest_api/v1/modules/?name=***module***&stream=***stream***&active=true

Making modules

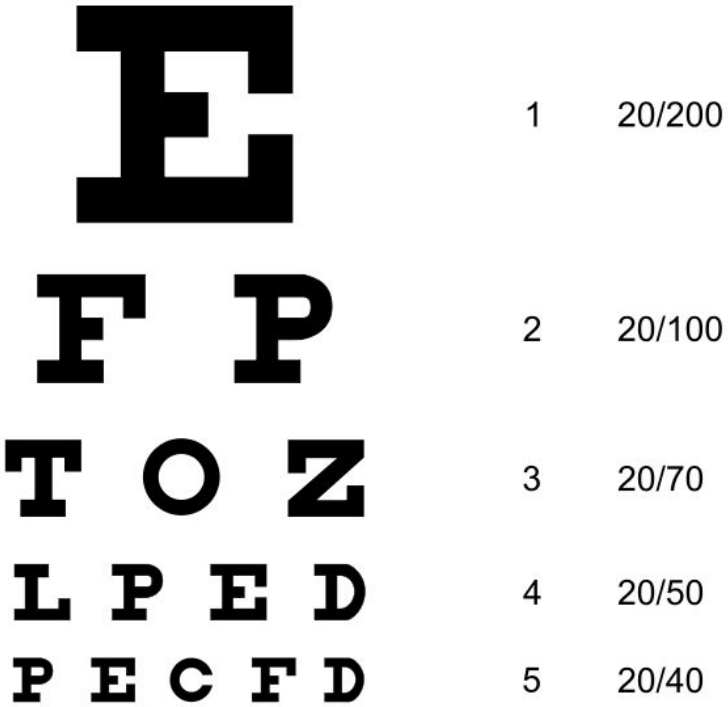
The modular way…

1. Create/update RPMs and modulemd.
2. Submit the module build.

The non-modular way…

1. Create/update RPMs.
2. Submit build for RPM #1 for F29.
3. Submit build for RPM #1 for F30.
4. Submit build for RPM #1 for F31.
5. Submit build for RPM #2 for F29.
6. Submit build for RPM #2 for F30.
7. Submit build for RPM #2 for F31.
8. ...

Example modulemd file



```
document: modulemd
version: 2
data:
    summary: An example module
    description: >-
        A module for the demonstration of the metadata format. Also,
        the obligatory lorem ipsum dolor sit amet goes right here.
    license:
        module:
            - MIT
    dependencies:
        - buildrequires:
            platform: []
          requires:
            platform: []
    api:
        rpms:
            - package-one
            - package-one-extras
            - package-one-cli
            - package-one-devel
            - package-two
    profiles:
        default:
            description: A standard installation.
            rpms:
                - package-one
                - package-one-extras
                - package-two
        cli:
            description: A command-line client.
            rpms:
                - package-one-cli
    components:
        rpms:
            first-package:
                ref: 3.0
                rationale: Provides the core functionality.
            second-package:
                ref: latest
                rationale: Web UI for the first-package.
```

Source control

Fedora dist-git “modules” namespace

https://src.fedoraproject.org/projects/modules/*

Repo name: ***modulename***

Branch name: ***stream***

File name: ***modulename***`.yaml`

The `fedmod` utility

```
fedmod rpm2module pkg1 [ pkg2 … ] > module.yaml

fedmod lint module.yaml
```

Module builds

- `fedpkg module-build`

Build monitoring and results via command line

- `fedpkg module-build-watch` ***`buildid`***
- `fedpkg module-build-info` ***`buildid`***

Build monitoring and results via web UI

- https://release-engineering.github.io/mbs-ui/

Using module builds

Rawhide

- Wait
- `dnf module …`

Older supported releases

- https://bodhi.fedoraproject.org/
- Wait
- `updates-testing-modular` repo enabled?
- `dnf module …`

Using module builds sooner

Install `odcs-client` package

`odcs create module` ***`module:stream:version:context`***

Local module builds

Online

- `fedpkg build-module-locally …`

Offline

- `fedpkg build-module-locally --offline \`
  `--repository` ***`repofile`*** `…`

Using local module builds

```
mod=$HOME/modulebuild/builds/module-module-stream-version/results
```

Then:

```
dnf --repofrompath=mymodule,$mod --enablerepo=mymodule module …
```

or create repo file in `/etc/yum.repos.d`:

```
dnf config-manager --add-repo $mod
dnf module …
```

Scratch module builds

`fedpkg module-scratch-build [ --file module.yaml ]`

Build monitoring and results via command line

- `fedpkg module-build-watch buildid`
- `fedpkg module-build-info buildid`

Build monitoring and results via web UI

- https://release-engineering.github.io/mbs-ui/

Using scratch module builds

```
mod=$HOME/mymodule ; mkdir $mod ; cd $mod
fedpkg module-build-info buildid
```

 # For each component Koji Task listed in the build info:

```
koji download-task taskid
```

 # For the module's Koji Tag listed in the build info, replace "+" with "%2b", then:

```
curl "https://pdc.fedoraproject.org/rest_api/v1/modules/?koji_tag= tag" \
  |  jq -r ".results[].modulemd" > modules.yaml
```

 # Manually remove any RPMs listed in the “filter” section of `modules.yaml`

```
createrepo_c $mod
modifyrepo_c --mdtype=modules modules.yaml $mod/repodata
```

 # Then use **`dnf --repofrompath=mymodule,$mod`** *… as shown earlier*

Resources

Fedora Modularity Home Page

- https://docs.pagure.org/modularity/

Fedora Modularity Documentation

- https://docs.fedoraproject.org/en-US/modularity/

fedmod

- https://pagure.io/modularity/fedmod

IRC

- #fedora-modularity on FreeNode