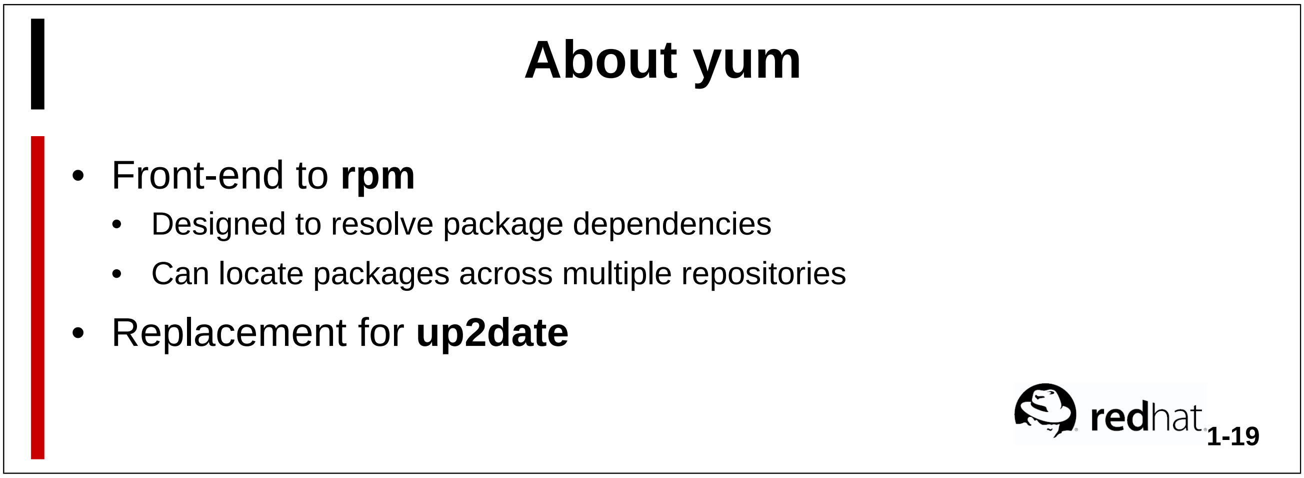


About yum

- Front-end to **rpm**
  - Designed to resolve package dependencies
  - Can locate packages across multiple repositories
- Replacement for **up2date**

redhat 1-19

Development of RPM cemented the future of Linux by greatly simplifying installation of software. As the operating system became more complex, RPM began to show a few weaknesses, primarily its inability to resolve dependencies.

```
[root@stationX ~]# rpm -ivh x3270-x11-*
warning: x3270-x11-3.3.4p7-3.el5.1.x86_64.rpm: Header V3 DSA signature: NOKEY, ↙
key ID 897da07a
error: Failed dependencies:
       x3270 = 3.3.4p7 is needed by x3270-x11-3.3.4p7-3.el5.1.x86_64
[root@stationX Server]#
```

Unfortunately, there is no way to look at the above output and determine if the suggested RPM, x3270, also has a dependency. As a matter of fact it is possible that dozens of RPMs would have to be installed. If the need RPMs were not available in the current directory, it would be up to the user to located and install each.

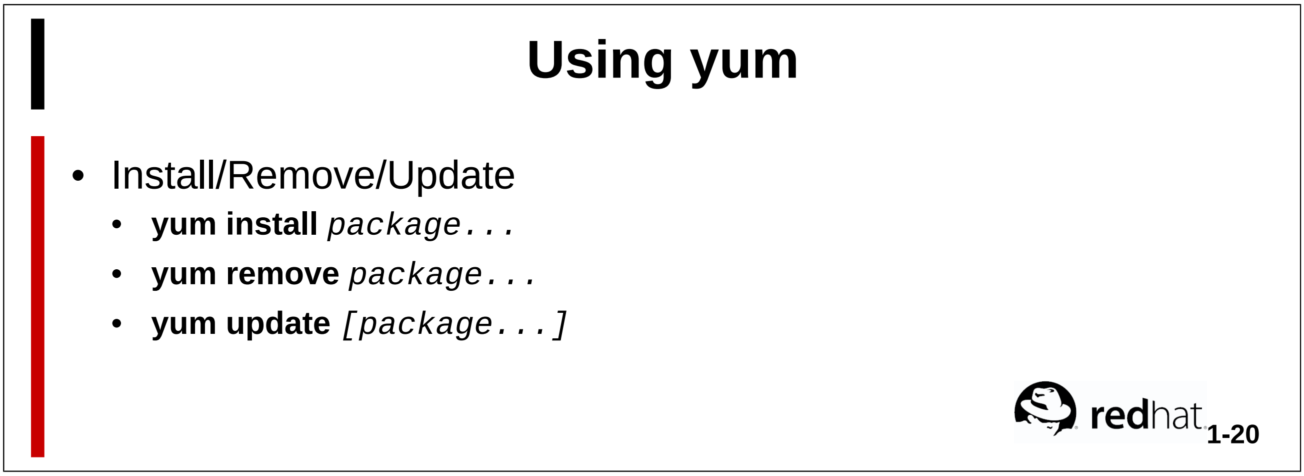
To solve the problem of dependency resolution and package location, volunteer programmers at Duke University developed Yellow dog Update, Modified, or YUM for short. The system is based on repositories that hold RPMs and a repodata filelist. The yum application can call upon several repositories for dependency resolution, fetch the RPMs, and install the needed packages.

```
[root@stationX ~]# yum install x3270-x11
... output truncated ...
Dependencies Resolved

=======================================================
Package                 Arch       Version          Repository        Size
=======================================================
Installing:
x3270-x11               x86_64     3.3.4p7-3.el5.1  server1           424 k
Installing for dependencies:
x3270                   x86_64     3.3.4p7-3.el5.1  server1           142 k

Transaction Summary
========================================================
Install      2 Package(s)
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 566 k
Is this ok [y/N]:
```



Using yum

- Install/Remove/Update
  - **yum install** *`package...`*
  - **yum remove** *`package...`*
  - **yum update** *`[package...]`*

1-20

The command-line utility **yum** gives you an easy way to manage the packages on your system:

```
[root@stationX ~]# yum install firefox
```

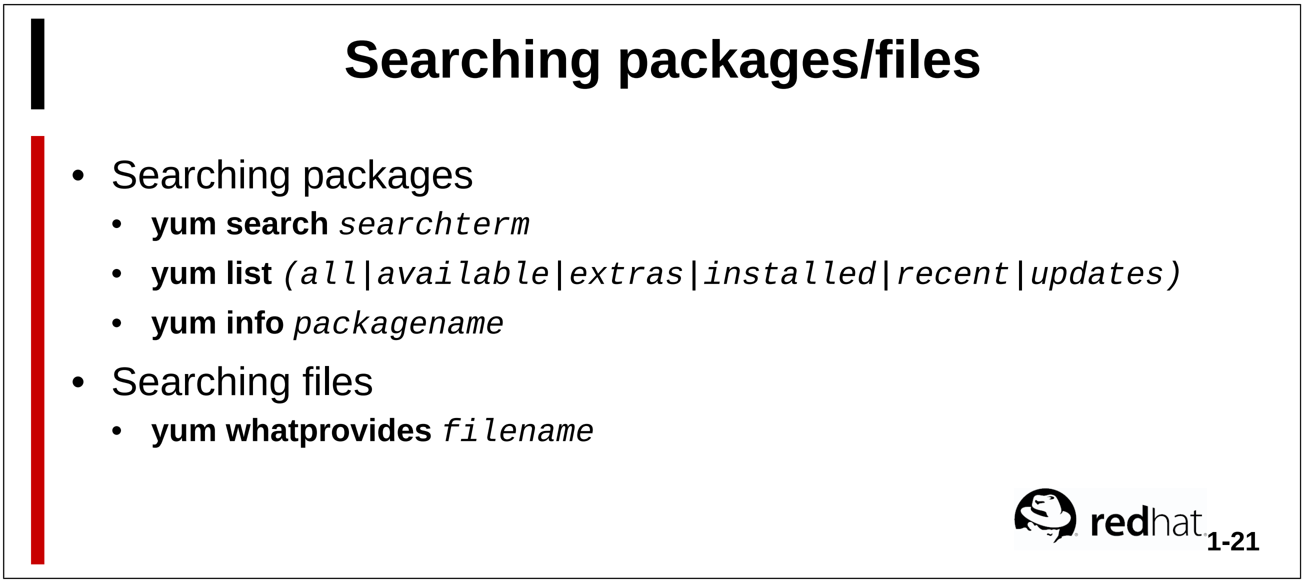
The above command will search the configured repositories for a package named firefox, and if found will install the latest version, pulling in dependencies if needed.

```
[root@stationX ~]# yum remove mypackage
```

The above command will try to remove the package named mypackage from your system. If any other package depends on mypackage **yum** will prompt you about this, giving you the option to remove those packages as well.

```
[root@stationX ~]# yum update [mypackage...]
```

If any packages are specified on the command-line **yum** will search the configured repositories for updated versions of those packages and install them. When no packages are specified **yum** will search for updates to all of your currently installed packages.



Searching packages/files

- Searching packages
  - **yum search** *searchterm*
  - **yum list** *(all|available|extras|installed|recent|updates)*
  - **yum info** *packagename*
- Searching files
  - **yum whatprovides** *filename*

1-21

yum search *searchterm* will search all known package names and descriptions for *searchterm*:

```
[root@stationX ~]# yum search cairo
```

yum list *searchterm* will search all known package names for *searchterm*. *searchterm* can include wildcards:

```
[root@stationX ~]# yum list '*irefo*'
```

yum info *package...* will search all the package database for *package* and display some info about the package.

```
[root@stationX ~]# yum info '*irefo*'
```

yum whatprovides *filename* will search all packages (both installed and available) for *filename*. This can be useful when you know the filename of an executable/library you need, but you don't know the package name.

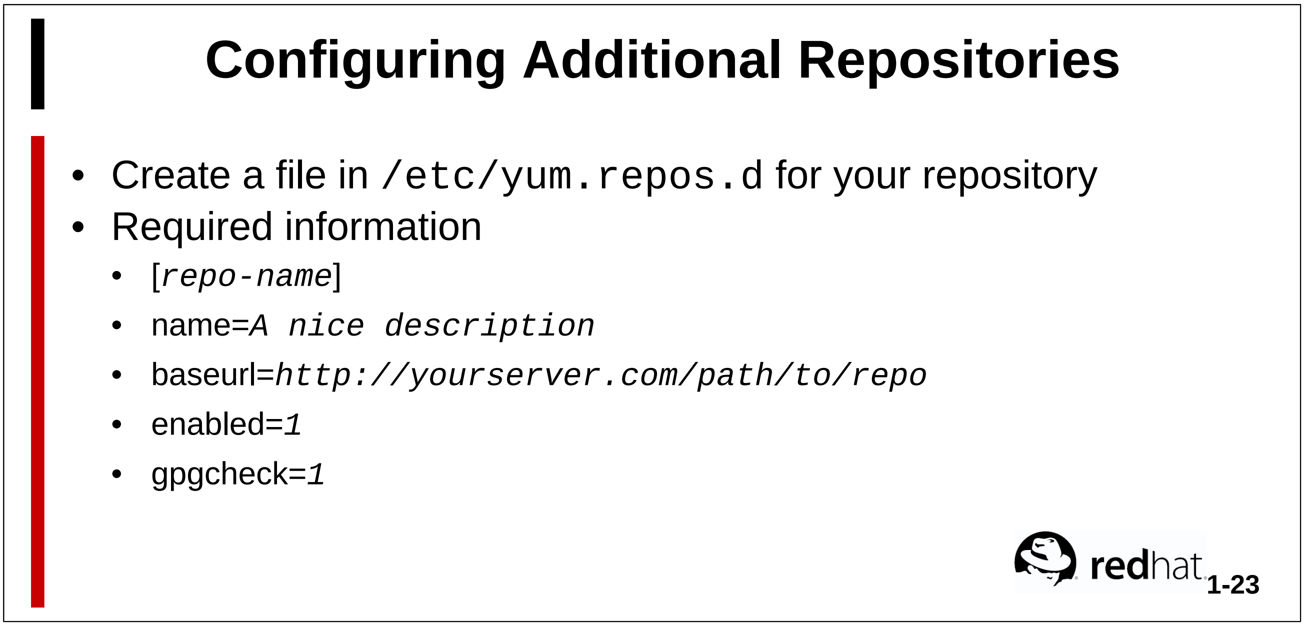
```
[root@stationX ~]# yum whatprovides /usr/sbin/sendmail
```

Configuring yum for a RHN Satellite Server

- Run **rhn_register** and select the satellite option
- Fill in the location of your satellite server
- Enter Account information

1-22

rhn_register will create configuration files for **yum** as well as registering your system with RHN or your satellite-server.

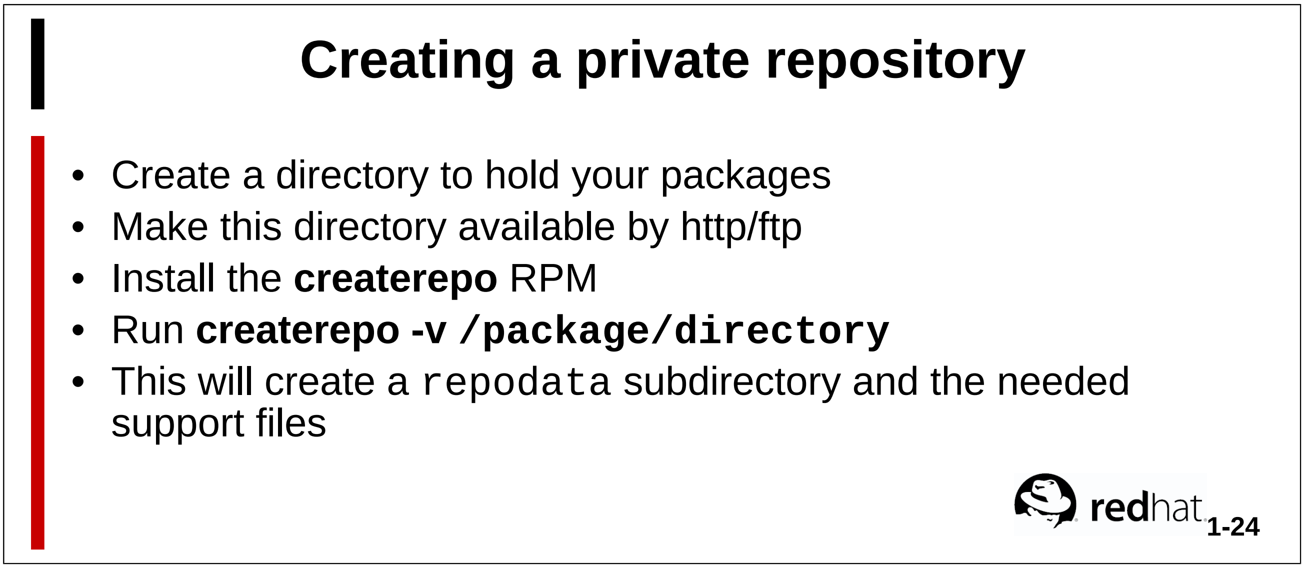


Configuring Additional Repositories

- Create a file in `/etc/yum.repos.d` for your repository
- Required information
  - [*repo-name*]
  - name=*A nice description*
  - baseurl=*http://yourserver.com/path/to/repo*
  - enabled=*1*
  - gpgcheck=*1*

1-23

Inside a repository declaration you can use variables like `$releasever` and `$basearch` to be substituted with the relevant information for your installation. Using this you can roll out one repository-file for use on multiple architectures or releases.



Creating a private repository

- Create a directory to hold your packages
- Make this directory available by http/ftp
- Install the **createrepo** RPM
- Run **createrepo -v `/package/directory`**
- This will create a `repodata` subdirectory and the needed support files

1-24

The **createrepo** creates the support files necessary for a yum repository. These files will put into the `repodata` subdirectory.

`repomd.xml` - The `repomd.xml` contains timestamps and checksum values for the other three files. Once a client has established a connection with a server, it caches all files, and only refreshes the cache if `repomd.xml` indicates the repo has changed.

`primary.xml.gz` - The `primary.xml.gz` file contains the list of all the RPMs in the repository, as well as dependency information. It also contains the information that would normally be returned by **rpm -qlp**.

`filelists.xml.gz` - The `filelists.xml.gz` file contains a list of all the files in all the RPMS. This is used by queries such as **yum whatprovides**.

`other.xml.gz` - The `other.xml.gz` file contains additional information, including the changelogs for the RPMs.

`comps.xml` - The optional `comps.xml` file contains information about package groups. This allows group installations and optimizes dependency resolution.

The addition or deletion of files within the repository requires a **createrepo** to be run again.