# Oracle Linux

## Managing Software in Oracle Linux

F19386-51
August 2025

Oracle Linux Managing Software in Oracle Linux,

F19386-51

# Contents

# Preface

Oracle Linux: Managing Software on Oracle Linux provides information about how to install, upgrade, and manage software on Oracle Linux systems by using DNF and Application Streams.

## Documentation License



## Conventions

The following text conventions are used in this document:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| monospace | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

## Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

## Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also

mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

# How Oracle Distributes Software Packages

Oracle uses two mechanisms to distribute software packages:

- Oracle Linux yum server
- Unbreakable Linux Network (ULN)

Depending on the infrastructure and the support agreement with Oracle, you can use either of these software distribution mechanisms with the Oracle Linux systems you're running.

You can also create software distribution mirrors to provision software to a broader infrastructure.

If you use Oracle Cloud Infrastructure (OCI), you can use the centralized OS Management Hub interface to manage software updates and patches for instances in OCI, private data centers, or supported third-party cloud environments.

> **Note**
>
> This guide covers software management using the Oracle Linux Yum Server only. Information about ULN has been moved to the Oracle Linux: Unbreakable Linux Network User's Guide and is relevant only to users who don't have access to OCI or OS Management Hub and who require access to restricted software that's not available on the Oracle Linux yum server.

2

# The Oracle Linux Yum Server

Instead of using the installation media, you can access the Oracle Linux yum server to install Oracle Linux packages, including bug fixes, security fixes, and enhancements. Oracle logically organizes software packages on the yum server into different repositories based on package purpose, support status, or dependencies.

## Available Oracle Linux Yum Servers

Two Oracle Linux yum sources for package distribution are available:

**Public Yum Server**
The primary Oracle Linux yum server is publicly available at https://yum.oracle.com/ where you can obtain software packages for free.
Other repositories that contain software, such as Ksplice, that are only licensed for use by Oracle Linux Support customers aren't available in the Oracle Linux yum server. For more information, see Available Yum Repositories.

**Oracle Cloud Infrastructure (OCI) Yum Servers**
Unlike the publicly available yum server, the Oracle Cloud Infrastructure (OCI) yum servers provide the complete range of software packages available on Unbreakable Linux Network (ULN), including Ksplice patch updates and other restricted software.
In addition, the ol10_oci_included, ol9_oci_included and ol8_oci_included yum repositories are also provided. The packages in these repositories must only be used on compute instances in OCI. The repositories are mirrored to all regional yum servers within OCI, but aren't mirrored to the publicly accessible Oracle Linux yum server.
To enable access to restricted content through the regional yum servers, ensure that you have installed the appropriate release-el8, release-el9, or release-el10 packages and have enabled the repositories to which you require access.

## Available Yum Repositories

A yum repository is a directory of packages that's typically available on a web server or an ISO image. The directory also includes metadata in a repodata subdirectory. The metadata is updated each time a package changes within the repository directory.

You can configure any client system to use a yum repository by creating a yum repository configuration entry. To install software from the repository, you use either the yum or dnf command to install software from the repository.

Yum repository names don't include the platform architecture because the URL to the repository already identifies the architecture. Therefore, when accessing the yum server, the system is automatically connected to the appropriate architecture's repositories.

Core OS repositories are the minimum required repositories for an Oracle Linux system to function. These repositories are enabled immediately after installation and must remain enabled through the life cycle of an Oracle Linux system.

## Oracle Linux 10

The following table lists the primary yum repositories for Oracle Linux 10. Other repositories are available on the Oracle Linux yum server. For a complete list, go to https://yum.oracle.com/oracle-linux-10.html.

| Repository | Description |
|---|---|
| ol10_baseos_latest | **Core repository**<br>Provides all the latest versions of the base OS packages in the current release of the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level. |
| ol10_appstream | **Core repository**<br>Provides all the latest versions of the Application Stream user space packages in the current release of the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level. |
| ol10_addons | Provides packages released by Oracle in addition to the upstream packages made available in the other repositories listed here. These packages are specific to functionality that Oracle provides to improve user experience on Oracle Linux and to provide access to services specific to Oracle. |
| ol10_codeready_builder | Provides the packages released in the upstream codeready_builder repository. The packages released in this repository are intended for developers who intend to build binary content from source packages. The packages include compilers, libraries, and source required for package building and other related tasks. Many of the packages in this repository have dependencies on packages in the ol10_appstream repository.<br>Support for the codeready_builder packages is limited to package installation help only. |

| Repository | Description |
|---|---|
| ol10_developer | Provides packages intended for developers to create test and development environments for Oracle Linux 10 and related technologies.<br>Support for the developer packages is limited to package installation help only. |
| ol10_developer_EPEL | Provides a mirror of the selected packages that are available on the EPEL (Extra Packages for Enterprise Linux) repository.<br>Support for the EPEL packages is limited to package installation help only. |

## Oracle Linux 9

The following table lists the primary yum repositories for Oracle Linux 9. Other repositories are available on the Oracle Linux yum server. For a complete list, go to https://yum.oracle.com/oracle-linux-9.html.

| Repository | Description |
|---|---|
| ol9_baseos_latest | **Core repository**<br>Provides all the latest versions of the base OS packages in the current release of the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level. |
| ol9_appstream | **Core repository**<br>Provides all the latest versions of the Application Stream user space packages in the current release of the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level. |
| ol9_addons | Provides packages released by Oracle in addition to the upstream packages made available in the other repositories listed here. These packages are specific to functionality that Oracle provides to improve user experience on Oracle Linux and to provide access to services specific to Oracle. |

| Repository | Description |
|---|---|
| ol9_codeready_builder | Provides the packages released in the upstream codeready_builder repository. The packages released in this repository are intended for developers who intend to build binary content from source packages. The packages include compilers, libraries, and source required for package building and other related tasks. Many of the packages in this repository have dependencies on packages in the ol9_appstream repository.<br>Support for the codeready_builder packages is limited to package installation help only. |
| ol9_developer | Provides packages intended for developers to create test and development environments for Oracle Linux 9 and related technologies.<br>Support for the developer packages is limited to package installation help only. |
| ol9_developer_EPEL | Provides a mirror of the selected packages that are available on the EPEL (Extra Packages for Enterprise Linux) repository.<br>Support for the EPEL packages is limited to package installation help only. |

## Oracle Linux 8

The following table lists the primary yum repositories for Oracle Linux 8. Other repositories are available on the Oracle Linux yum server. For a complete list, go to https://yum.oracle.com/oracle-linux-8.html.

| Repository | Description |
|---|---|
| ol8_baseos_latest | **Core repository**<br>Provides all the latest versions of the base OS packages in the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level. |
| ol8_appstream | **Core repository**<br>Provides all the latest versions of the Application Stream user space packages in the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level. |

| Repository | Description |
|---|---|
| ol8_baseos_base | Provides the base versions of the base OS packages in the distribution when a particular update level is released. For the initial release of Oracle Linux 8, *n* has a value of 0. Errata patches aren't provided in this repository. If the system has the _baseos_latest repository enabled, you don't need to enable this repository to keep the system updated and secure. |
| ol8_u*n*_baseos_patch | Provides the patched versions of the base OS packages in the distribution when a particular update level is released. As errata patches are made available, the updates are released in this repository. Note that in the case of the initial release of Oracle Linux 8, *n* has a value of 0. Errata patches are provided in this repository until a new update release is made available. To keep a system updated and secure, enable the appropriate _baseos_latest repository. If the system has the _baseos_latest repository enabled, you don't need to enable a patch repository. |
| ol8_addons | Provides packages released by Oracle in addition to the upstream packages made available in the other repositories listed here. These packages are specific to functionality that Oracle provides to improve user experience on Oracle Linux and to provide access to services specific to Oracle. |
| ol8_codeready_builder | Provides the packages released in the upstream codeready_builder repository. The packages released in this repository are intended for developers who intend to build binary content from source packages. The packages include compilers, libraries, and source required for package building and other related tasks. Many of the packages in this repository have dependencies on packages in the ol8_appstream repository.<br>Support for the codeready_builder packages is limited to package installation help only. |
| ol8_developer | Provides packages intended for developers to create test and development environments for Oracle Linux 8 and related technologies.<br>Support for the developer packages is limited to package installation help only. |
| ol8_developer_EPEL | Provides a mirror of the selected packages that are available on the EPEL (Extra Packages for Enterprise Linux) repository.<br>Support for the EPEL packages is limited to package installation help only. |

# Securing the Distribution of Oracle Linux Packages

Oracle Linux yum servers are configured to use HTTPS so that all communications are validated, verified, and encrypted during package download.

Oracle Linux packages are signed by using Gnu Privacy Guard (GnuPG or GPG) key pairs. You can check package veracity by using the public keys that we provide to authenticate that the packages come from Oracle and that they haven't been altered since they were signed.

The system's repository files for Oracle Linux packages are normally set up with GPG parameters so that GPG verification is completed automatically as part of the download process. For example, the following entry in /etc/yum.repos.d/oracle-linux-ol9.repo is configured to automatically use the appropriate GPG key to verify the package during download:

```
[ol9_baseos_latest]
name=Oracle Linux 9 BaseOS Latest ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL9/baseos/latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
...
```

Where:

- gpgkey: specifies the full path of the key that's provided by the repository maintainer.
- gpgcheck=1: the default 1 setting indicates that package installation automatically uses the GPG key to verify the packages to be installed are trusted packages. Always ensure that gpgcheck=1 is the persistent setting.

The public keys that Oracle generates for Oracle Linux packages are available on the Oracle Linux yum server and are included when the packages are installed on the system. The public GPG key is installed automatically when you install the oraclelinux-release package.

> **Note**
>
> Developer Preview packages might be signed using a dedicated development GPG key. The development GPG key isn't installed on Oracle Linux systems by default, so you might need to install the key and manually verify such packages.

For more information, and download links for other Oracle Linux release keys and checksum files, see https://linux.oracle.com/security/gpg/

# About the DNF Utility

The dnf utility, which is based on Dandified Yum (DNF), is the client software for installing and managing packages on systems running Oracle Linux 8 or later releases. While installing or upgrading packages, dnf also automatically handles package dependencies and requirements.

DNF provides significant improvements in functionality and performance when compared to the traditional yum command. DNF also brings a host of new features, including modular content,

and a more stable and documented API. DNF is compatible with Yum v3 for editing or creating configuration files and for managing repositories and packages. You can use the dnf command and all its options in the same manner as how you use the yum command on previous releases of Oracle Linux.

To provide backward compatibility, the yum and dnf commands are interchangeable. You not only can perform tasks similar to those that you performed in earlier releases of Oracle Linux, but you can also avail of a wider range of new features that are available in dnf, such as improved package management and performance. To view syntax differences between dnf and legacy yum commands, see Comparing Yum Version 3 With DNF.

When you run the dnf command, the system connects to the yum server repository and downloads the latest software packages to the system in RPM format. The dnf command then displays a list of the available packages so that you can select which packages you want to install.

**Important**

Oracle Linux packages are built as RPM packages. However, avoid using the rpm command for install or update operations unless explicitly instructed to do so by a support representative. In particular, if you do use the rpm command, never use the --force or --nodeps options. Otherwise, you might cause serious system stability issues.

For more information, see the dnf(8) manual page and https://dnf.readthedocs.io/en/latest/index.html.

3

# Configuring a System to Use Oracle Linux Yum Server

Define global configuration options in the dnf.conf file. Then edit, or create .repo files to define which repositories to subscribe to.

## Configuring the Global DNF Configuration Settings

Global DNF configuration settings are configured in the main configuration section in /etc/dnf/dnf.conf.

The global definitions for DNF are found under the [main] section heading of the DNF configuration file. The following table lists important directive for DNF.

> **Note**
>
> For backward-compatibility purposes, a symbolic link to /etc/dnf/dnf.conf is created at /etc/yum.conf. The configuration syntax is the same; although, some configuration options have been deprecated and some new configuration options have been added. See Comparing Yum Version 3 With DNF for a list of the differences between configuration options and syntax.

See the dnf.conf(5) manual page for more information.

| Directive | Description |
|---|---|
| cachedir | Directory used to store downloaded packages. |
| debuglevel | Logging level, from 0 (none) to 10 (all). |
| exclude | A space separated list of packages to exclude from installs or updates, for example: exclude=VirtualBox-7.? kernel*. |
| gpgcheck | If set to 1, verify the authenticity of the packages by checking the GPG signatures. You might need to set gpgcheck to 0 if a package is unsigned, but be wary that the package could have been maliciously altered. |
| gpgkey | Path to the GPG public key file. |
| installonly_limit | Maximum number of versions that can be installed of any one package. |
| keepcache | If set to 0, remove packages after installation. |
| logfile | Path to the dnf log file. |
| obsoletes | If set to 1, replace obsolete packages during upgrades. |
| plugins | If set to 1, enable plugins that extend the functionality of dnf. |

| Directive | Description |
|---|---|
| proxy | URL of a proxy server including the port number. See Configuring a System to Use a Proxy With a Yum Server |
| proxy_password | Password for authentication with a proxy server. |
| proxy_username | Username for authentication with a proxy server. |
| reposdir | Directories where dnf looks for repository files with a .repo extension. The default directory is /etc/yum.repos.d. See Subscribing to Different Yum Repositories. |

**Example [main] Configuration**

The following listing shows an example [main] section from the DNF configuration file.

```
[main]
cachedir=/var/cache/dnf
keepcache=0
debuglevel=2
logfile=/var/log/dnf.log
obsoletes=1
gpgkey=file://media/RPM-GPG-KEY
gpgcheck=1
plugins=1
installonly_limit=3
```

# Configuring a System to Use a Proxy With a Yum Server

If the organization uses a proxy server as an intermediary for internet access, specify the proxy setting in /etc/dnf/dnf.conf as shown in the following example.

```
proxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, also specify the proxy_username, and proxy_password settings.

```
proxy=http://proxysvr.example.com:3128
proxy_username=user
proxy_password=password
```

> ⚠ **Caution**
>
> All dnf users require read access to /etc/dnf/dnf.conf or /etc/sysconfig/rhn/up2date. If these files must be world-readable, don't use a proxy password that's the same as any user's login password, and especially not root's password.

# Configuring Access to the Oracle Linux Yum Server Through a Firewall

The Oracle Linux yum server delivers content through a Content Delivery Network (CDN). When you connect to the Oracle Linux yum server, you connect to a node on the CDN that's geographically closer to the system you're using for faster download speeds.

Because the CDN has many nodes that run on different networks around the world, you can not configure any single IP address or network range for an egress firewall rule.

In environments where a strict firewall policy limits outbound connections for systems, we recommend working with the OS Management Hub service and setting up a management station within a demilitarized zone (DMZ) on the network. Otherwise, configure a local yum mirror server within a DMZ. See Setting Up a Local Yum Mirror for information on how to configure a local yum mirror.

# Subscribing to Different Yum Repositories

Subscribe to different yum repositories in Oracle Linux.

Oracle Linux uses modular yum repository configuration files that are made available as release packages that are maintained through yum. Release packages simplify repository management and also ensures that yum repository definitions are kept up-to-date automatically whenever you update the system.

On all Oracle Linux systems, the oraclelinux-release-el*n* package is installed by default. This package contains the core repository configurations to access all the repositories required for an Oracle Linux system to install common OS software packages and the other release packages used to obtain other yum repository configurations.

To install the yum repository configuration for a particular set of software, use the dnf command to install the corresponding package.

1. List the available release configuration packages.

   To view a list of available RPM files for managing yum repository configurations, run the following command.

- Oracle Linux 10
- Oracle Linux 9
- Oracle Linux 8

## Oracle Linux 10

```
dnf list "*release-el10*"
```

## Oracle Linux 9

```
dnf list "*release-el9*"
```

### Oracle Linux 8

```
dnf list "*release-el8*"
```

2. Install any required yum repository configuration packages.

   Use the dnf command to install the corresponding package for any required yum repository configurations. For example, to install the available yum configurations for developer release software on Oracle Linux, you can run:

   - Oracle Linux 10
   - Oracle Linux 9
   - Oracle Linux 8

   ### Oracle Linux 10

   ```
   sudo dnf install oraclelinux-developer-release-el10
   ```

   ### Oracle Linux 9

   ```
   sudo dnf install oraclelinux-developer-release-el9
   ```

   ### Oracle Linux 8

   ```
   sudo dnf install oraclelinux-developer-release-el8
   ```

## Editing Yum Repository Configuration Files

DNF uses yum repository configuration files to configure where to install different packages and their dependencies from. By default, repository configuration files are stored in the /etc/yum.repos.d directory. You can define another directory location to store repository configurations by setting the reposdir directive in the dnf.conf file.

Use the repository directory to define .repo files for repositories that you want to make available. A .repo file can contain entries for more than one yum repository. To subscribe to a repository, you can edit the enabled option to a value of 1 and save the configuration file. The change has immediate effect.

The following table describes the basic directives for a repository. Any other directive that appears in the repository file override the corresponding global definition in the [main] section of the DNF configuration file. See the dnf.conf(5) manual page for more information.

| Directive | Description |
|---|---|
| baseurl | Location of the repository (expressed as a file://, ftp://, http://, or https:// address). This directive must be specified. |
| enabled | Whether to enable dnf to use the repository. Set the value to 1 to enable the repository, or 0 to disable the repository. |
| name | Descriptive name for the repository. This directive must be specified. |



## Oracle Linux 10

The following listing shows an example repository section from a .repo configuration file.

```
[ol10_appstream]
name=Oracle Linux $releasever Application Stream ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL10/appstream/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

In this example, the values of gpgkey and gpgcheck override any global setting. dnf substitutes the name of the current system's architecture for the variable *$basearch*.

## Oracle Linux 9

The following listing shows an example repository section from a .repo configuration file.

```
[ol9_appstream]
name=Oracle Linux $releasever Application Stream ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL9/appstream/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

In this example, the values of gpgkey and gpgcheck override any global setting. dnf substitutes the name of the current system's architecture for the variable *$basearch*.

## Oracle Linux 8

The following listing shows an example repository section from a .repo configuration file.

```
[ol8_appstream]
name=Oracle Linux $releasever Application Stream ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL8/appstream/$basearch
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

In this example, the values of gpgkey and gpgcheck override any global setting. dnf substitutes the name of the current system's architecture for the variable *$basearch*.

## Configure Compute Instances Access to Regional Yum Server Repositories

Compute instances in Oracle Cloud Infrastructure (OCI) have access to regional yum servers through the service gateway. The base URL of the repository uses the $ociregion variable to define which regional server to use and the $ocidomain variable to define the domain where the yum server is located. By using variables, configuration can remain relatively standard across Oracle Linux deployments but provide access to extra resources available to OCI customers.

For example, the base URL to the ol8_baseos_latest repository for Oracle Linux 8 is:

```
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL8/baseos/latest/$basearch
```

You can set the $ociregion variable by populating content in /etc/dnf/vars/ociregion. For example, if $ociregion is set to -phx, the base URL expands to point to the regional yum server in Phoenix.

Typically, when you create an instance, this value is set to point to the closest regional yum server on the OCI service network. If the /etc/dnf/vars/ociregion file doesn't exist, or the file is empty, the base URL points to the publicly accessible Oracle Linux yum server.

## Using the DNF config-manager Plugin

The dnf config-manager plugin is used to manage DNF configuration and yum repositories.

The dnf-plugins-core package includes several utilities that can help you to manage configuration and safely apply updates to existing configuration. The most significant of these utilities is the dnf config-manager plugin, which can be used to edit and manage DNF configuration.

The following options are available:

**--add-repo *url***
You can use dnf config-manager to add repositories, either at a specified URL or within a specified repository file. For example, to add a repository configuration file for Oracle Linux that's hosted on a remote server, you can run the following command:

```
sudo dnf config-manager --add-repo https://example.com/my_yum_config.repo
```

You can use the same command to automatically generate a repository configuration file for a valid yum repository by pointing to the URL of which the repository is hosted. For example, to create a configuration file in /etc/repos.d for an example repository, run the following command:

```
sudo dnf config-manager --add-repo https://example.com/repo/el-release/myrepo/x86_64
```

**--enable *repo_name***
To enable a repository, use the --enable option. For example, to enable a repository named *myrepo*, run the following command:

```
sudo dnf config-manager --enable myrepo
```

**--disable *repo_name***
To disable a repository, use the --enable option. For example, to disable a repository named *myrepo*, run the following command:

```
sudo dnf config-manager --disable myrepo
```

**--setopt *option=value***
Use the --setopt option to set a configuration option to a specified value. For example, to disable GPG key validation on a repository named *myrepo*, run the following command:

```
sudo dnf config-manager --save --setopt=gpgcheck=0 myrepo
```

Note that the --save option is often used with --setopt, so that the value is saved to the configuration.

**--save**
Saves the current configuration.

**--dump**
To view the current DNF configuration, use the --dump option to output the configuration to stdout.

See the dnf.plugin.config_manager(8) manual page for more information.

# How to Recover the Base Yum Repository Configuration

Perform this task if the system's base repository configuration has been corrupted or otherwise lost.

1. Create a temporary repository configuration file in /etc/yum.repos.d/temp_base.repo, and populate the file with entries corresponding to the system's OS version.

- Oracle Linux 10
- Oracle Linux 9
- Oracle Linux 8

## Oracle Linux 10

For Oracle Linux 10:

```
sudo tee /etc/yum.repos.d/temp_base.repo <<EOF
[ol10_baseos_latest]
name=Oracle Linux 10 BaseOS Latest (\$basearch)
baseurl=https://yum\$ociregion.\$ocidomain/repo/OracleLinux/OL10/baseos/latest/\$basearch/
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

### Oracle Linux 9

For Oracle Linux 9:

```
sudo tee /etc/yum.repos.d/temp_base.repo <<EOF
[ol9_baseos_latest]
name=Oracle Linux 9 BaseOS Latest (\$basearch)
baseurl=https://yum\$ociregion.\$ocidomain/repo/OracleLinux/OL9/baseos/latest/\$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
EOF
```

### Oracle Linux 8

For Oracle Linux 8:

```
sudo tee /etc/yum.repos.d/temp_base.repo <<EOF
[ol8_baseos_latest]
name=Oracle Linux 8 BaseOS Latest (\$basearch)
baseurl=https://yum\$ociregion.\$ocidomain/repo/OracleLinux/OL8/baseos/latest/\$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

2. Reinstall the required release packages of the system's OS version to set up the standard yum repository configurations.

- Oracle Linux 10
- Oracle Linux 9
- Oracle Linux 8

### Oracle Linux 10

```
sudo dnf reinstall oraclelinux-release-ol10
```

### Oracle Linux 9

```
sudo dnf reinstall oraclelinux-release-ol9
```

### Oracle Linux 8

```
sudo dnf reinstall oraclelinux-release-ol8
```

3. Verify that the recovery is successful.

- Oracle Linux 10
- Oracle Linux 9
- Oracle Linux 8

### Oracle Linux 10

```
ls /etc/yum.repos.d/oraclelinux-release-ol10
```

### Oracle Linux 9

```
ls /etc/yum.repos.d/oraclelinux-release-ol9
```

### Oracle Linux 8

```
ls /etc/yum.repos.d/oraclelinux-release-ol8
```

4. Remove the temporary configuration file.

```
rm /etc/yum.repos.d/temp_base.repo
```

5. Reinstall other required release packages to obtain the correct repository configurations.

```
sudo dnf reinstall repository
```

6. Enable the repositories that you need.

```
sudo dnf config-manager --enable repository
```

4

# Installing Software on Oracle Linux

Software packages are installed on a system by using standard dnf commands and depend on the system having the appropriate yum repositories enabled.

Use the dnf install command to install a package and any of its dependencies:

```
sudo dnf install package_name
```

> **Note**
>
> dnf makes no distinction between installing and upgrading a kernel package. If you run dnf install for a package that's already installed on the system, and a newer version of the package is available, the command upgrades the package. You can use the dnf upgrade command, or the synonymous dnf update command, to upgrade all packages installed on the system. See Updating Software on Oracle Linux.

The system notifies you of any extra packages that might be installed and prompts you to confirm whether to go ahead with the installation. For example:

```
Last metadata expiration check: 1:38:46 ago on Mon 14 Apr 2025 13:47:07 BST.
Dependencies resolved.
================================================================================
 Package            Architecture   Version                 Repository            Size
================================================================================
Reinstalling:
 dnf-automatic      noarch         4.14.0-17.0.1.el9       ol9_baseos_latest     51 k

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

Total download size: 51 k
Installed size: 54 k
Is this ok [y/N]
```

You can bypass the confirmation check in a dnf install command by using the -y option.

For a list of dnf commands that are commonly used to manage DNF packages and repositories, see DNF Command References.

**Related Topics**

- Subscribing to Different Yum Repositories
  Subscribe to different yum repositories in Oracle Linux.

# Managing Sets of Packages Using DNF Groups

Sets of packages can be organized and managed as a *group* to provide everything that you might need for a particular function.

Examples include the groups for setting up a virtualization host, a graphical desktop, a collection of fonts, or core system administration tools.

Groups contain listings of packages that are required as mandatory for the base functionality of the group, default packages that are installed for the group on most systems, and optional packages that you might decide to install for extra functionality.

Groups can also be nested so that a parent group contains a set of subgroups that can be installed.

For a list of `dnf` commands that are commonly used to manage DNF groups, see Table 7-2 of DNF Command References.

- To list all package groups, use the `dnf group list` command.

  For example, to list all groups with their respective ID values, run:

  ```
  dnf group list --ids
  ```

  You can use the `--installed`, `--available`, and `--hidden options` to list particular group types.

- To view group info, such as any subgroups or packages, use the `dnf group info` command.

  To view the information for the Core group, which contains the minimal number of packages that can be installed on a system, run:

  ```
  dnf group info core
  ```

  Note that groups can contain subgroups. For example, the "Server with GUI" group contains many subgroups. When you use the `dnf group info` command on these parent groups, the subgroups are listed. You might need to run the command again on each subgroup to finally see the package listings for each subgroup.

- To install a group package, use the `dnf group install` command.

  For example, to install the default packages from the "Server with GUI" group, run:

  ```
  sudo dnf group install "Server with GUI"
  ```

  You can also use the group id with the group install command:

  ```
  sudo dnf group install graphical-server-environment
  ```

  Use the `--with-optional` option to also install optional packages from the group.

  To limit the packages installed for a group to only the mandatory package list, you need to set the DNF group default package types temporarily, for example:

  ```
  sudo dnf --setopt=group_package_types="mandatory" group install graphical-server-environment
  ```

# Using DNF Modules and Application Streams

DNF introduces the concepts of modules, streams, and profiles for managing different versions of software applications within a single OS release. Modules can be used to group many packages that consist of a single application and its dependencies. Streams can be used to provide alternative versions of the same module. Profiles can be used to define optional configurations of any single module so that a module can be limited only to developer packages or can be scoped to include other packages for enhanced functionality.

Modular content is made available separate to core OS packages in Oracle Linux 8 and Oracle Linux 9 so that these user-space applications can be installed in various user-space environments, including virtual machines, containers, and the base OS. Modular content for Oracle Linux 8 and Oracle Linux 9 is typically shipped within the Application Stream (AppStream) repository.

> **Important**
>
> **No modular packages are available for Oracle Linux 10**. Different versions of userspace packages continue to be available as Application Streams but don't use the package modularity available in previous releases.

For a list of application stream packages in the latest Oracle Linux version, see Oracle Linux: Product Life Cycle Information.

- **Modules**: Are a set of RPM packages that are grouped together and must be installed together. They can contain several streams that consist of several versions of applications that you can install. You enable a module stream to provide system access to the RPM packages that are contained in that module stream.

  A typical module can contain the following types of packages:

  - Packages with an application
  - Packages with the application's specific dependency libraries
  - Packages with documentation for the application
  - Packages with helper utilities

- **Module streams**: Hold different versions of content contained within a module.

  Modules can have several streams, where each stream contains a different version of packages and their dependencies. Each stream receives updates independently. A module can have more than one stream. However, note that for each module, only one of its streams can be enabled to provide access to its packages. Often, the stream with the latest version is selected as the default stream and is used when operations don't specify a particular stream or a different stream hasn't been enabled before.

  Module streams can be thought of as virtual repositories within the physical repository. For example, the `postgresql` module provides the PostgreSQL database in streams 9.6 and 10, with version 10 being the current default stream.

> **Note**
>
> We recommend that you use the latest stream for any module that's installed, even though other streams might continue to receive limited support.

- *Module profiles*: Provide a list of certain packages that are to be installed at the same time for a particular use case. At the same time, profiles are also a recommendation by the application packagers and experts. Note that each module can have one or more profiles.

  You install packages by using a module's profile as a one-time action. Using a module's profile to install packages doesn't prevent you from installing or uninstalling any of the packages that are provided by the module. Furthermore, you can install packages by using the profiles of the same module without any further preparation. Also, a module's package list can contain packages from outside of the module stream. These packages are often from BaseOS or the stream's own dependencies. Note that modules in Application Stream always have a default profile. This default profile is used for installations, when no other profile is explicitly specified.

  For example, The `httpd` module that includes the Apache web server includes the following profiles for installation:

  - `common`: This profile is a hardened production-ready deployment and is the default profile.
  - `devel`: This profile installs the packages that are necessary to make modifications to `httpd`.
  - `minimal`: This profile installs the smallest set of packages that provide a running web server.

Unlike software collections that were included in previous releases of Oracle Linux, applications that are installed from Application Streams are installed into standard locations and don't require extra commands or actions to run. You can run any version of an installed application the same way as any other version, regardless of the stream from which it was installed. After it's installed, the application behaves exactly as any other native application that you have installed by using DNF.

To manage modules, you use a combination of `dnf` and `dnf module` subcommands. For a list of `dnf` commands that are commonly used with modules, see Table 7-3 of DNF Command References.

## About Modular Dependencies and Stream Changes

Typically, packages that provide content depend on other packages and specify the appropriate dependency versions. This same mechanism also applies to packages that are contained within modules. The grouping of packages and their particular versions into modules and streams has some extra constraints. For example, module streams can declare dependencies on the streams of other modules, independent of the packages that the modules contain and provide. After any package or module operation, the entire dependency tree for all the underlying installed packages must satisfy all the conditions that the packages declare. Likewise, all the module stream dependencies must satisfied.

These extra constraints require that you analyze and understand the implications before performing any package operation. Changing the enabled module streams doesn't automatically manipulate packages to enable you to have complete control over the changes. However, the tool always provides a summary of the actions to take.

When performing package operations on modules and streams, keep the following guidelines, caveats, and warnings in mind:

- Enabling a module stream might also require the enabling of streams of other modules.
- Installing a module stream profile or packages from a stream might also require the enabling of streams of other modules and the installation of extra packages.
- Disabling a stream of a module might also require the disabling of other module streams, as no packages are removed automatically.
- Removing a package can require the removal of other packages. If any of the packages are provided by modules, the module streams remain enabled in preparation for further installation, even if no packages from these streams are installed later; thereby, mirroring the behavior of an unused yum repository.
- Switching the stream that's enabled for a module is the same as resetting the current stream and enabling a new stream.

> **Note**
>
> Switching an enabled stream doesn't automatically change any of the installed packages. Also, removing packages that are provided by a previous stream, and any of the packages that depend on them, and the installation of packages in a new stream are all tasks that must be performed manually.

- Because of potential upgrade scripts that run during an installation, directly installing a stream of a module, other than one that's already installed by default, isn't recommended.

Module dependencies include regular package dependencies that are similar to RPM dependencies. For modules, however, availability can also depend on the enabling of module streams; module streams can also depend on other module streams.

Dependencies of non modular packages on modular packages are used in Application Stream only when a modular package is provided by a module stream that's marked as the default. When a modular package depends on a non modular package, the system always retains the module and stream choices, unless you provide explicit instructions to change them. A modular package receives updates from the enabled stream of the module that provides this package and doesn't upgrade to a version from a different stream.

# Enabling Modules

Selected modules are available by default when you install Oracle Linux. You can install these modules as needed. Note that directly installing a stream of a module, other than one that's installed by default, might cause unexpected issues.

This task uses the example of the PHP module to show you how to enable modules.

1. Check the status of the module.

```
sudo dnf module list php
```

```
Oracle Linux 8 Application Stream (x86_64)
Name        Stream        Profiles                    Summary
php         7.2 [d]       common [d], devel, minimal  PHP scripting language
php         7.3           common [d], devel, minimal  PHP scripting language
php         7.4           common [d], devel, minimal  PHP scripting language
```

```
php          8.0          common [d], devel, minimal          PHP scripting language

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

Each PHP version has 3 corresponding profiles, with the common profile as the default. The [d] flag indicates the default version of the profile.

2. List the packages that are installed for the profiles associated with the module stream.

```
sudo dnf module info --profile php:8.0
```

The output displays each profile and its associated packages that are required for the module. For example:

```
Name    : php:8.0:8060020211215065547:0a326c83:x86_64
common  : php-cli
        : php-common
        : php-fpm
        : php-mbstring
        : php-xml
...
```

3. Enable the module stream you want to use.

```
sudo dnf module enable php:8.0 -y
```

4. Optionally, check the status of the selected stream to verify that it has the e flag.

```
sudo dnf module list php
```

```
Name         Stream         Profiles                    Summary
php          7.2 [d]        common [d], devel, minimal  PHP scripting language
php          7.3            common [d], devel, minimal  PHP scripting language
php          7.4            common [d], devel, minimal  PHP scripting language
php          8.0 [e]        common [d], devel, minimal  PHP scripting language
```

5. To switch to a preferred module stream, see Switching Module Streams.

# Removing Installed Modules

Removing an installed module removes all the packages and their dependencies that are installed by the profiles of the enabled module stream.

The modules to be removed must have some profiles already installed.

Before removing an installed module, review the information in About Modular Dependencies and Stream Changes.

This task only removes packages that are listed in profiles of the operational module stream. Packages that aren't listed in any of module stream's profiles remain installed on the system and can be removed manually. At each step, a summary of pending changes is first displayed and you're prompted to confirm the action before proceeding.

1. Remove the module.

   sudo dnf module remove *module-name*

2. Disable the module stream.

   sudo dnf module disable *module-name*

3. Remove any packages that you manually installed from the module stream.

   sudo dnf remove *package* ...

## Switching Module Streams

By switching module streams, you can obtain a different version of the current module that's being used in the system.

The module stream that you want to switch must already be enabled *and*, at the same time, another stream of the same module must already exist.
Before switching module streams, review the information in About Modular Dependencies and Stream Changes.

In some steps, a summary of pending changes is first displayed and you're prompted to confirm the action before proceeding.

You can also use the dnf module switch-to command for this task. See Table 7-3.

1. Reset the module to enable you to install an alternative stream.

   sudo dnf module reset *module-name*

2. Install the profiles of a different stream of the module.

   sudo dnf install @*module-name*:*stream*

   You might also need to apply changes to other module streams and packages.

3. Update or downgrade any packages installed from the previous module stream that weren't listed in the profiles installed in the previous step.

   sudo dnf distro-sync

4. Manually remove any remaining packages that were installed from the previous module stream.

   sudo dnf remove *package* ...

5

# Updating Software on Oracle Linux

Software updates are achieved using standard dnf commands on the system and depend on the system having the appropriate yum repositories enabled.

You can use the dnf install and dnf update commands to handle general package installation or updates.

To update a system to use the latest packages that are available, run:

```
sudo dnf upgrade
```

> **Note**
>
> The dnf update command automatically runs dnf upgrade.

Update the system often to ensure that packages have the latest security patches and bug fixes. Consider using automatic updates so that software is correctly maintained on the system.

Note than if you use the dnf install command for software that's already installed, the software packages that you specify are also updated to the latest available version.

After performing a system update where many packages are updated, we recommend that you reboot the system. System functionality might become unstable if core packages are updated and the system isn't restarted to load the most recent updates. You can check whether a system requires a restart by running:

```
dnf needs-restarting -r
```

## Updating Software Automatically

The DNF Automatic tool is provided as an extra package that you can use to keep the system automatically updated with the latest security patches and bug fixes. The tool can provide notifications of updates, download updates, and then install them automatically by using systemd timers.

See the dnf-automatic(8) manual page for more information.

> **Important**
>
> By using Oracle Ksplice, you can keep an Oracle Linux kernel patched and updated all the time, without any need to reboot. For Oracle Linux Support customers, Ksplice is an essential tool to keep the systems safe, secure, and updated. See Oracle Linux: Ksplice User's Guide for more information.

1. Install the dnf-automatic package.

   ```
   sudo dnf install -y dnf-automatic
   ```

2. Enable the systemd dnf-automatic.timer timer unit to start using the service.

   ```
   sudo systemctl enable --now dnf-automatic.timer
   ```

   Optionally run other timer units to override the default configuration. Often, these timer units are used as handy shortcuts to perform a specific behavior:

   - dnf-automatic-notifyonly.timer: Notifies for available updates
   - dnf-automatic-download.timer: Downloads package updates, but doesn't install them
   - dnf-automatic-install.timer: Downloads and automatically installs package updates

   You enable the required behavior by running:

   ```
   sudo systemctl enable --now dnf-automatic-install.timer
   ```

3. Configure the DNF Automatic tool by editing the /etc/dnf/automatic.conf configuration file and then restarting the timer unit.

# Disabling Updates for Particular Packages

To disable updates for specific packages, add an exclude statement to the [main] section of the /etc/dnf/dnf.conf file. See Configuring the Global DNF Configuration Settings. For example, to exclude updates for VirtualBox and kernel:

```
exclude=VirtualBox-7.? kernel*
```

> **Note**
>
> Excluding certain packages from being updated can cause dependency errors for other packages. A system could also become vulnerable to security-related issues if you don't install the latest updates.

# Tracking Security Updates and Errata Releases

Oracle releases important changes to the Oracle Linux software as individual package updates, known as errata.

Errata packages can contain the following:

- Security advisories, which have names prefixed by ELSA-* (for Oracle Linux) and OVMSA-* (for Oracle VM).
- Bug fix advisories, which have names prefixed by ELBA-* and OVMBA-*.
- Feature enhancement advisories, which have names prefixed by ELEA-* and OVMEA-*.

To be notified when new errata packages are released, you can subscribe to the Oracle Linux and Oracle VM errata mailing lists at https://oss.oracle.com/mailman/listinfo/el-errata and https://oss.oracle.com/mailman/listinfo/oraclevm-errata.

You can track updates to Oracle Linux yum server repositories by visiting https://yum.oracle.com/whatsnew.html, where you can see which packages were updated within each repository for the previous six months.

> **Note**
>
> Oracle doesn't comment on existing security vulnerabilities except through Errata announcements. To provide the best security posture to all Oracle customers, Oracle fixes significant security vulnerabilities in severity order. So, the most critical issues are always fixed first. Fixes for security vulnerabilities are produced in the following order:
>
> - Latest code line refers to the code being developed for the next major Oracle release of the product.
> - Next patch set for all non terminal releases

## Using DNF to See Security Updates

DNF includes integrated options to handle any requirement for managing security and errata updates that are available for packages installed in Oracle Linux.

For more information, see the dnf(8) manual page.

- List available errata and security updates.

  List the errata that are available for a system as follows:

  ```
  sudo dnf updateinfo list
  ```

  The output from the command sorts the available errata in order of their IDs and identifies their types, which can be one of the following:

  - Security patch (*severity*/Sec.)
  - Bug fix (bugfix)
  - Feature enhancement (enhancement)

  Security patches are also listed according to their severity, which can be Critical, Important, Moderate, or Low.

  - Use the --sec-severity option to filter the security errata by severity, for example:

    ```
    sudo dnf updateinfo list --sec-severity=Critical
    ```

  - To list the security errata by their Common Vulnerabilities and Exposures (CVE) IDs instead of their errata IDs, specify the keyword cves as an argument:

    ```
    sudo dnf updateinfo list cves
    ```

    Similarly, the keywords bugfix, enhancement, and security filter the list for all bug fixes, enhancements, and security errata.

  - You can use the --cve option to display the errata that correspond to a specific CVE ID, for example:

    ```
    sudo dnf updateinfo list --cve CVE-2022-3545
    ```

- Display more information about a security updated.

  To display more information about a CVE, specify info instead of list, for example:

  ```
  sudo dnf updateinfo info --cve CVE-ID
  ```

- Perform security related updates on a system.

  Use any of the following options:

  - To update all the packages for which security-related errata are available to the latest versions of the packages, even if those packages that include bug fixes or new features but not security errata, use the following command:

    ```
    sudo dnf --security update
    ```

  - To update all packages to the latest versions that contain security errata, ignoring any newer packages that don't contain security errata, use the following command:

    ```
    sudo dnf --security upgrade-minimal
    ```

  - To update all kernel packages to the latest versions that contain security errata, use the following command:

    ```
    sudo dnf --security upgrade-minimal kernel*
    ```

  - To update only those packages that correspond to a CVE or erratum, use the dnf update --cve command. For Enterprise Linux Security Advisory (ELSA) patches, use dnf update --advisory.

    ```
    sudo dnf update --cve CVE-ID
    ```

    ```
    sudo dnf update --advisory ELSA-ID
    ```

> **Note**
>
> Some updates might require that you reboot the system. By default, the boot manager automatically enables the most recent kernel version.

6

# Using Software Distribution Mirrors

Distribution mirrors are alternative sources of software packages to repositories on the Oracle Linux yum server. These are selected repositories that you locally replicate from the public server. The local repositories become the package sources for client systems that exist in the local network.

Distribution mirrors are useful in complex infrastructures and are important when developing a controlled update strategy for a mission critical production environment. Distribution mirrors are deployed to provide the following services:

- Provide access to yum repositories for systems that don't have access to a public network.
- Improve software download times and reducing bandwidth overhead for larger infrastructure
- Set up network-based installation infrastructure
- Cater for a snapshot style update strategy where testing can be performed against a controlled software distribution environment before the updates are implemented on production systems.

A server that functions as a software distribution mirror contains yum repositories. The repositories can be made available to client systems in the internal network through various methods such as using local web server, a file transfer server, and so on.

The software distribution mirror must be synchronized with the official Oracle Linux sources. If required, you can control synchronization to occur at strategic intervals so that you can test system updates against a known set of package versions before you roll them out to all the infrastructure.

> **Note**
>
> If you use Oracle Cloud Infrastructure (OCI), you can use the centralized OS Management Hub interface to manage software updates and patches for instances in OCI, private data centers, or supported third-party cloud environments. The management station that's configured for on-premises or third-party cloud instances acts as a local yum and DNF mirror of any configured Oracle Linux repositories.

## Prerequisites for the Local Distribution Mirror

The system that you set up as a local distribution mirror must meet the following criteria:

- Must have Internet access to connect to the official Oracle Linux sources.
- Must have at least 6 GB of memory to create the yum metadata.
- Must be configured to provide access to the mirrored repositories by system clients.
- Must have enough disk space to store copies of the packages that it hosts.

  When calculating for the needed disk space, consider the following:

- Disk space requirements depend on the number and size of the repositories that you decide to mirror. Other factors are the number of clients to be serviced, including their platforms, OS, and other specific packages that each client might be using and which would require updates.
- Disk space that's used for a mirror is only consumed and is never released. Thus, disk requirements aren't static and can increase over time.
- Packages in the repositories are also updated on a regular schedule and these updates affect the storage requirements on the local yum server.

For guidance in estimating the disk size requirements, run the following command:

```
sudo dnf repoinfo [repo-ID]
```

Part of the command output includes the size of a specific repository, for example:

```
...
Repo-id       : ol9_baseos_latest
...
Repo-size     : 54 G
...

Repo-id       : ol9_addons
...
Repo-size     : 2.3 G
...
```

Because repositories are dynamic and grow over time, always plan to allocate substantially greater disk space than what Repo-size specifies. Optionally, you can also create a dedicated file system and mount this to the directory that hosts the mirrored repositories.

# Setting Up a Distribution Mirror

Systems can be configured to distribute packages and provide updates to client systems within a local network without the need for clients to access the public servers through the Internet.

You can select any method to provide access to the local repositories in the mirror server. This task uses HTTP as an example.

1. Ensure that the latest version of the yum-utils is installed on the system.

   ```
   sudo dnf install -y yum-utils
   ```

2. Install the Apache HTTP server.

   ```
   sudo dnf install -y httpd
   ```

3. Create a base directory for the local repositories, for example:

   ```
   sudo mkdir -p /var/www/html/yum
   ```

4. If you created a dedicated file system for the mirror, mount that file system to the base directory. For example, edit the /etc/fstab with the details of the file system mounted on the /var/www/html/yum, then run the following:

   ```
   sudo mount device-path
   ```

   In the previous example, device-path is the path to the device. For example:

   ```
   sudo mount /dev/sda
   ```

5. Check if SELinux is enabled. For example, the following command shows that SELinux is enabled and set to Enforcing:

   ```
   sestatus
   SELinux status:             enabled
   ...
   Current mode:               enforcing
   Mode from config file:      enforcing
   ...
   ```

6. If SELinux is enabled in enforcing mode and you have mounted another file system on /var/www/html/yum/ , do the following steps:

   a. Define the default file type of the repository root directory hierarchy as httpd_sys_content_t. For example, if the base directory is /var/www/html/yum/, then run the following:

      ```
      sudo /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/www/html/yum(/.*)?"
      ```

   b. Apply the file type to the entire repository.

      ```
      sudo /sbin/restorecon -R -v /var/www/html/yum/
      ```

7. Edit the HTTP server configuration file, /etc/httpd/conf/httpd.conf, as follows:

   a. Specify the resolvable domain name or the IP address of the server in the argument to ServerName.

      ```
      ServerName system-mirror:80
      ```

   b. Verify that in the <Directory "/var/www/html"> section, the setting of the Options directive specifies Indexes and FollowSymLinks, for example:

      ```
      Options Indexes FollowSymLinks
      ```

      With this setting, you can browse the directory hierarchy.

8. Start the HTTP server and configure it to start after a reboot.

   ```
   sudo systemctl start httpd
   sudo systemctl enable httpd
   ```

9. If you enabled a firewall on the system, configure it to enable incoming HTTP connection requests on TCP port 80.

```
sudo firewall-cmd --add-service=http
sudo firewall-cmd --permanent --add-service=http
```

10. Choose how you want the local mirror to function to serve clients in the local network. See: Set up the system as a yum mirror.

# Setting Up a Local Yum Mirror

A system that functions as a local yum repository mirrors repositories from the public Oracle Linux yum server.

When Oracle Linux is installed on this system, that system automatically contains the repositories that are required by the system's OS. These repositories are found in the system's /etc/yum/repos.d directory. The repositories are defined in different /etc/yum/repos.d/*.repo files.

By mirroring these default repositories, the system can function as a local yum server to service clients that have the same OS and platform as the mirror.

However, you might want the local yum mirror to also service clients that use different OS releases for other platforms. In this case, you would need to define other repositories that are required by those clients.

## Configuring the Local Yum Server

Setting up a system to function as a local yum server involves mirroring required repositories from the public Oracle Linux yum server.

The yum mirror must meet the requirements described in Prerequisites for the Local Distribution Mirror. Also, you must have completed the procedure in Setting Up a Distribution Mirror.

You can mirror any repository available on the Oracle Linux yum server, if you have the definition for the repository configured in /etc/yum.repos.d. Mirroring repositories that the system already has available is uncomplicated. However, for other repositories, you might need to be more specific about the which repositories you want to mirror. Moreover, you might need to provide other repository configuration.

1. Mirror all the current system's enabled repositories to the base directory.

```
sudo dnf reposync --delete --download-metadata -p /var/www/html/yum
```

**--delete**
Remove from the mirror any package that's removed upstream. Using this option is highly recommended.

**--download-metadata**
Include all repository metadata in the synchronization.

If you run the command for the first time, the process might take a long while to complete. At the end of the process, the system becomes ready to provide packages to client systems with compatible OS and platforms as the mirror.

2. Set the local mirror to host repositories for heterogeneous clients.

   a. Create the required repositories for mixed clients.

      Suppose that the server is running the latest Oracle Linux 10 release, but must provide packages for Oracle Linux 9 and Oracle Linux 8 clients. You would do the following:

      - Create /etc/yum.repos.d/9-mirror.repo with entries similar to the following example:

        ```
        [ol9_baseos_latest]
        name=Oracle Linux 9 BaseOS Latest  ($basearch)
        baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL9/baseos/latest/$basearch/
        gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
        gpgcheck=1
        enabled=0

        [ol9_appstream]
        name=Oracle Linux 9 Application Stream Packages ($basearch)
        baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL9/appstream/$basearch/
        gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
        gpgcheck=1
        enabled=0
        ```

      - Create /etc/yum.repos.d/8-mirror.repo with entries similar to the following example:

        ```
        [ol8_baseos_latest]
        name=Oracle Linux 8 BaseOS Latest  ($basearch)
        baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL8/baseos/latest/$basearch/
        gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
        gpgcheck=1
        enabled=0

        [ol8_appstream]
        name=Oracle Linux 8 Application Stream Packages ($basearch)
        baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL8/appstream/$basearch/
        gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
        gpgcheck=1
        enabled=0
        ```

      Full yum configurations for different releases are available at https://yum.oracle.com/mirror/. Navigate to the correct distribution and architecture and download the appropriate *.repo files.

      > **Important**
      >
      > All entries must have enabled=0 to prevent packages from these repositories to be installed on the local yum mirror itself.

b. Mirror each repository in the *.repo files to the base directory.

```
sudo dnf reposync --repoid ol9_baseos_latest --delete --download-metadata -p /var/www/html/yum
...

sudo dnf reposync --repoid ol8_baseos_latest --delete --download-metadata -p /var/www/html/yum
...
```

3. Automate the regular client package updates from mirrored repositories through a cron script or systemd timer unit.

   For example, create a file at /etc/cron.daily/yum-mirror-update with the following content:

```
#!/bin/bash
# Regularly update yum repos
dnf reposync --delete --download-metadata -p /var/www/html/yum
```

   Ensure that the file is executable.

```
sudo chmod +x /etc/cron.daily/yum-mirror-update
```

   - If the yum mirror services mixed clients, change the script to resemble the following:

```
#!/bin/bash

REPOS=(repo-IDs)

for REPO in ${REPOS[@]}
do
   dnf reposync --repo=$REPO --delete --download-metadata -p /var/www/html/yum
done
```

     *repo-IDs* represents a comma separated list of the IDs of repositories that are required by **all** the clients of the mirror. These IDs are contained in corresponding /etc/yum.repos.d/*.repo files you created for those clients. In this procedure's example, the *repo-IDs* would represent Oracle Linux 10 repositories for clients that are compatible with the mirror. In addition, you would include ol9_baseos_latest, ol9_appstream, ol8_baseos_latest, ol8_appstream, and so on, for the other clients.

4. Configure clients appropriately to access these repositories.

   See Configuring Client Access to the Local Mirror.

# Using rsync to Mirror the Oracle Linux Yum Server

Oracle provides an rsync interface to the Oracle Linux yum server repositories at the yum-rsync.oracle.com domain that maps directly to the URL structure of the Oracle Linux yum server.

You must fulfill the requirements as described in Prerequisites for the Local Distribution Mirror. Also, you must complete the procedure as provided in Setting Up a Distribution Mirror.

With the rsync interface, you can easily mirror the Oracle Linux yum server for broader usage without any requirement for complex system configuration. This approach is helpful for large enterprises that want to mirror entire repository structures for all architectures. The rsync

interface is an alternative method to running the reposync command to synchronize mirrored repositories.

1. Install rsync on the system.

   ```
   sudo dnf install -y rsync
   ```

2. Use rsync to mirror all the repositories that you intend to mirror.

   For example, to mirror all the Oracle Linux 10 repositories for all architectures, you can recursively mirror everything at the rsync://yum-rsync.oracle.com/repo/OracleLinux/OL10/ endpoint.

   ```
   rsync -arv rsync://yum-rsync.oracle.com/repo/OracleLinux/OL10 /var/www/html/yum/
   ```

   You can mirror a particular repository for a particular architecture by providing a more specific URL. For example, to mirror the current Oracle Linux 9 baseos repository for the x86_64 architecture, you would type:

   ```
   mkdir -p /var/www/html/yum/OL9/baseos/latest
   rsync -arv rsync://yum-rsync.oracle.com/repo/OracleLinux/OL9/baseos/latest/x86_64 /var/www/html/yum/OL9/
   baseos/latest/
   ```

## Mirroring Repositories From an ISO

The local yum mirror can be configured to mirror repositories from an ISO image to make them available to clients.

You must fulfill the requirements as described in Prerequisites for the Local Distribution Mirror. You must also complete the procedure as provided in Setting Up a Distribution Mirror.

This task assumes that you're mirroring the repositories from an Oracle Linux 10 image. It also assumes that to provide access to the mirror, you're using a web server.

1. Mount the ISO image at an appropriate location so you can copy its contents.

   ```
   sudo mount -o loop,ro OL10.iso /mnt
   ```

2. Create a directory to host the repositories from the ISO.

   ```
   sudo mkdir -p /var/www/html/yum/10_ISO
   ```

3. Copy the repositories from the ISO to the new directory.

   ```
   sudo cp -r /mnt/BaseOS /var/www/html/yum/10_ISO/
   sudo cp -r /mnt/AppStream /var/www/html/yum/10_ISO/
   ```

4. Configure clients appropriately to access these repositories.

   See Configuring Client Access to the Local Mirror.

# Configuring Client Access to the Local Mirror

Clients require access to the local repository mirror to receive updates and errata fixes.

A local mirror must be configured where the clients connect. See previous sections in Using Software Distribution Mirrors.

Perform this task on all the clients in the local network.

1. Import the GPG key.

   ```
   sudo gpg --import /etc/pki/rpm-gpg/RPM-GPG-KEY
   ```

   The location of the GPG key might differ depending on the Oracle Linux release that's installed on the system. You can also download and import the GPG keys directly from the Oracle Linux yum server. See https://yum.oracle.com/faq.html#a10 for more information.

2. Disable any existing yum repositories configured in the /etc/yum.repos.d directory.

   Select from one of the following methods:

   - Edit each /etc/yum.repos.d/*.repo file to specify an enabled=0 setting for each entry in the file.
   - Perform a global disable operation.

     ```
     sudo dnf config-manager --disable \*
     ```

   - Remove the .repo extension from the file names to cause yum operations to ignore these files.

     ```
     sudo bash -c "cd /etc/yum.repos.d;
     for i in *.repo;
       do mv $i $i.disabled;
     done"
     ```

3. Create a local *.repo file, such as /etc/yum.repos.d/local-yum.repo, and populate it with repository entries from the local mirror.

   **Tip**

   To distinguish the local repositories from the public yum repositories, prefix the names of their entries with a string such as local_.

- Oracle Linux 10
- Oracle Linux 9
- Oracle Linux 8

## Oracle Linux 10

The following example shows entries for an Oracle Linux 10 client:

```
[local_ol10_baseos_latest]
name=Oracle Linux 10 BaseOS Latest ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol10_appstream]
name=Oracle Linux 10 Application Stream ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol10_addons]
name=Oracle Linux 10 Addons ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

## Oracle Linux 9

The following example shows entries for an Oracle Linux 9 client:

```
[local_ol9_baseos_latest]
name=Oracle Linux 9 BaseOS Latest ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol9_appstream]
name=Oracle Linux 9 Application Stream ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol9_addons]
name=Oracle Linux 9 Addons ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

## Oracle Linux 8

The following example shows entries for an Oracle Linux 8 client:

```
[local_ol8_baseos_latest]
name=Oracle Linux 8 BaseOS Latest ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol8_appstream]
name=Oracle Linux 8 Application Stream ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol8_addons]
name=Oracle Linux 8 Addons ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol8_ksplice]
name=Ksplice for Oracle Linux 8  ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[local_ol8_userspace_ksplice]
name=Ksplice-aware user space packages for Oracle Linux 8 ($basearch)
baseurl=http://local_mirror/repo-location/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

For *local_mirror*, you can specify either the local server's resolvable host name or its IP address.

Ensure that the following configurations are correct:

- All the entries have an `enabled=1` setting.
- The `baseurl` points to the correct mirror location that contains the repositories that each client requires. The locations depend on how you organized the repositories in the mirror's base directory, such as `/var/www/html/yum`.
- The correct GPG key file must exist at the path that's specified for the `gpgkey` parameter. You can download the GPG keys used to sign all the Oracle Linux release packages from the Oracle Linux yum server. See https://yum.oracle.com/faq.html#a10 for more information.

4. Test the configuration.

a. Clear the yum metadata cache.

```
sudo dnf clean metadata
```

b. Verify that the relevant repositories are listed for the client.

```
sudo dnf repolist
```

If the client can not connect to the local yum server, check that the firewall settings on the local yum server enable incoming TCP connections to the HTTP port, which is typically port 80.

5. After confirming that the correct repositories are configured on the client, obtain updates from the local server.

```
sudo dnf update
```

# 7
# DNF Command References

The following tables show examples of some common tasks that you can perform by using the dnf command to manage packages and package groups.

**Table 7-1 DNF Commands**

| Command | Description |
|---|---|
| dnf repolist | Lists all the enabled repositories. |
| dnf list | Lists all the packages that are available in all enabled repositories and all packages that are installed on the system. |
| dnf list installed | Lists all the packages that are installed on the system. |
| dnf list available | Lists all the packages that are available to be installed in all enabled repositories. |
| dnf search *string* | Searches the package descriptions for the specified string. |
| dnf provides *feature* | Finds the name of the package to which the specified file or feature belongs, for example:<br>dnf provides /etc/dnf/automatic.conf |
| dnf info *package* | Displays detailed information about a package, for example:<br>dnf info dnf-automatic |
| dnf repoquery -l *package* | List the files that are contained in a package and are installed when the package is installed, for example:<br>dnf repoquery -l dnf-automatic |
| dnf install *package* | Installs the specified package, including packages on which it depends, for example:<br>dnf install dnf-automatic |
| dnf check-update | Checks whether updates exist for packages that are already installed on the system. |
| dnf upgrade *package* | Updates the specified package, including packages on which it depends, for example:<br>dnf upgrade dnf-automatic<br>DNF also interprets the dnf update *package* command as synonymous with the upgrade syntax; however, this syntax is considered deprecated. |
| dnf upgrade | Updates all packages, including packages on which they depend.<br>DNF also interprets the dnf update *package* command as synonymous with the upgrade syntax; however, this syntax is considered deprecated. |

**Table 7-1 (Cont.) DNF Commands**

| Command | Description |
|---|---|
| dnf remove *package* | Removes the specified package. For example:<br>dnf remove dnf-automatic |
| dnf clean all | Removes all cached package downloads and cached headers that contain information about remote packages. Running this command can help clear problems that are a result of unfinished transactions or out-of-date headers. |
| dnf help | Displays help about dnf usage. |
| dnf help *command* | Displays help about the specified dnf command, for example:<br>dnf help upgrade |
| dnf shell | Runs the dnf interactive shell. |

**Table 7-2 DNF Group Commands**

| Command | Description |
|---|---|
| dnf group list | Lists Environment Groups, that contain many subgroups; and base groups of packages that are available for installation.<br>To include hidden groups in the list and all the groups' IDs, add the --hidden -v options. |
| dnf group info *groupname* | Displays detailed information about a group. If the group is a parent group, this command lists all subgroups that it contains, alternately the command lists all packages that are in the group.<br>To include the groups' IDs, use the -v option. |
| dnf group install *groupname* | Installs all the packages in a group. |
| dnf group update *groupname* | Updates all the packages in a group. |
| dnf group remove *groupname* | Removes all the packages in a group. |

**Table 7-3 DNF Module Commands**

| Command Syntax | Description of Action | Additional Information |
|---|---|---|
| dnf install *package* | Installs the specified package. | If a package is provided by a module stream, the dnf command resolves the required module stream and enables it automatically during package installation. In addition, the process is recursive for any package dependencies. Note that if more module streams satisfy the requirement, the default streams are used.<br>If the package is provided by a module stream that isn't marked as default or isn't enabled, that package isn't recognized until you manually enable the applicable module stream. |
| dnf module enable*module-name*:*stream* | Enables a module or stream. | Use this command when you want to enable a module so that the packages are available to the system, but you don't necessarily want to install the module immediately.<br>Note that some modules might not define default streams. In this case, you must explicitly specify the stream. If you explicitly specify a stream and an alternative stream is set as the default, the enabled stream overrides the default stream for later install requests.<br>As an alternative, you can use dnf module switch-to. See the entry in the current table. |
| dnf install *@module-name*<br>Alternatively, you can use:<br>dnf module install *module-name* | Installs a module. The @ character is shorthand to indicate that you intend to install a module. | If the module defines a default stream, or you have enabled a particular stream, you don't need to include *stream* and *colon* in the command syntax.<br>Be aware that some modules don't define default streams.<br>As an alternative, you can use dnf module switch-to. See the entry in the current table. |
| dnf install*@module-name*:*stream*<br>Alternatively, you can use:<br>dnf module install*module-name*:*stream* | Installs a module by using a specific stream and default profiles. | As an alternative, you can use dnf module switch-to. See the entry in the current table. |

**Table 7-3 (Cont.) DNF Module Commands**

| Command Syntax | Description of Action | Additional Information |
|---|---|---|
| dnf install@*module-name*:*stream*/*profile*<br>Alternatively, you can use:<br>dnf module install*module-name*:*stream*/*profile* | Installs a module by using a specific stream and profile. | As an alternative, you can use dnf module switch-to. See the entry in the current table. |
| dnf module info *module-name* | Displays information about a module. | |
| dnf module info --profile *module-name* | Displays information about the packages that are installed by the profiles of a module using the *default* stream. | |
| dnf module info --profile*module-name*:*stream* | Displays information about the packages that are installed by the profiles of a module using a *specified* stream. | |
| dnf module list | Lists all the available modules and displays the module name, stream, profiles, and a summary. Each module and stream is listed on a separate line. Profiles are indicated using comma separated values for each module and stream.<br>Default values are indicated with the characters [d]. Modules that are enabled are indicated with the characters [e], while those that are disabled are indicated with the characters [x]. Installed modules, streams, and profiles are indicated with the characters [i]. | |
| dnf module list *module-name* | Lists the current status of a module. | |
| dnf module provides *package* | Displays information about which modules provide a specified package.<br>If the package is only available outside of any modules, the command output is empty. | |

**Table 7-3 (Cont.) DNF Module Commands**

| Command Syntax | Description of Action | Additional Information |
|---|---|---|
| dnf module switch-to@*module-name*:*stream*/*profile* | Switch to a module stream. This command also changes the versions of installed packages to those versions provided by the new stream and removes packages from the old stream that are no longer available. The command also updates installed profiles if these are available in the new stream. | This command is more encompassing than dnf module enable command because it both enables modules and runs distrosync on all modular packages in the enabled modules.<br>The command is also more encompassing than dnf module install because it both installs profiles and runs distrosync on all modular packages in the installed module. However, you must specify a profile with this command because it doesn't use default profiles. |
| dnf module remove @*module-name* | Removes the specified module. | Removing an installed module removes all the packages and their dependencies that are installed by the profiles of the enabled module stream.<br>The modules to be removed must have some profiles already installed. |
| dnf module remove --all@*module-name*:*stream* | Removes all packages from the specified stream. | Note that the command can remove critical packages from the system. |
| dnf module remove@*module-name*:*stream*/*profile* | Removes packages from the installed profile. | |
| dnf module remove@*module-name*:*stream* | Removes packages from all installed profiles within the specified stream. | |
| dnf module reset @*module-name* | Resets a module to the initial state so it's no longer enabled or disabled. So, all installed profiles are removed. | Note that the command doesn't remove packages from the specified module. |
| dnf module disable @*module-name* | Disables a module and all its streams.<br>All related module streams become unavailable. So, all installed profiles are removed. | Note that the command doesn't remove packages from the specified module. |

8

# Comparing Yum Version 3 With DNF

The following table compares Yum v3 features, commands, and options with the DNF tool that's introduced in Oracle Linux 8.

| Yum v3 Feature, Command, or Option | DNF Feature, Command, or Option | Notable Differences |
|---|---|---|
| --skip-broken option | --skip-broken option<br>Is an alias for the --setopt=strict=0 option | **When used for installations:** Skips all packages (or those with broken dependencies that are passed to DNF) without raising an error or causing the operation to fail.<br>You can use either option with DNF. You can also set this behavior as the default in the dnf.conf file.<br>**When used for upgrades:** The semantics that were used to trigger the yum command with the--skip-broken option are set for dnf update as the default. Note that you don't need to use the --skip-broken option with the dnf upgrade command. Instead, use the --best option to use only the latest version of packages in transactions. |
| yum update command | dnf update command | Command syntax change only. No differences with the behavior for dnf update andyum update. |
| yum upgrade command | dnf upgrade command | Aside from the syntactical difference, the behavior of dnf upgrade is the same as yum upgrade. Note that in Yum v3, yum upgrade is the same as yum --obsoletes update. |
| clean_requirements_on_remove option | clean_requirements_on_remove option | This option is enabled by default in DNF, which might cause confusion when comparing the remove operation results between the two Yum versions, as DNF removes more packages. |
| resolvdep command | Not available<br>Use the dnf provides command to find which package provides a specific file. | The Yum v3 command is maintained for legacy purposes *only*. |

| Yum v3 Feature, Command, or Option | DNF Feature, Command, or Option | Notable Differences |
|---|---|---|
| deplist command | Not available<br>Use the dnfrepoquery--deplist command to find dependencies for a package. | The yumdeplist alias is provided for Yum v3 compatibility with the dnf repoquery --deplist command. |
| Excludes (and repository excludes) | Excludes (and repository excludes) | Yum v3 respects excludes during installations and upgrades; whereas, DNF respects all operations, including erasing, and listing. |
| includepkgs option | include option | In DNF, the directive name for repository (and main) configuration has been renamed for better alignment with its DNF counterpart, exclude . |
| skip_if_available option | skip_if_available option | This option is enabled by default in DNF.<br>Without this setting, and without explicitly setting skip_if_unavailable=True in the relevant repository .ini file, Yum immediately stops and reports a repository error. |
| overwrite_groups option | Not available | This configuration option has been removed in DNF. Instead, when DNF identifies several groups with the same group ID, it merges the contents of the groups. |
| mirrorlist_expire option | Not available | DNF uses metadata_expire for the expiring metadata, and the mirrorlist file. |
| "metalink" mention in the mirrorlist repository option. | Not available | A fix has been applied in DNF to render the following information in the yum.conf(5) inapplicable:<br>If the mirrorlist URL contains the word *metalink*, then the value of mirrorlist is copied to metalink (if metalink isn't set). |
| alwaysprompt option | Not available | This option has been removed from DNF to simplify configuration. |
| group_package_types option | Not available | This option has been removed from DNF to simplify configuration. |

| Yum v3 Feature, Command, or Option | DNF Feature, Command, or Option | Notable Differences |
|---|---|---|
| upgrade_requirements_on_install | Behaves as though disabled. | Because DNF tolerates the use of other package managers, it's possible that not all changes that are made to RPMDB are stored in the history of transactions. Thus, DNF doesn't fail in this situation, which means the force option is no longer required. |
| yum swap command | dnfshell command<br>This command performs a remove and install transaction.<br>dnf --allowerasing command | Using the dnf --allowerasing command is the same as using yum swap A B, where you want to replace A (providing P) with B (also providing P), which conflicts with A, without removing C (which requires P). |
| Dependency processing details displayed during the depsolving phase. | Not available | In DNF, the depsolver considers all dependencies for update candidates, which would result in a lengthy output. Note that the Yum v3 output can also be confusing and lengthy, especially for large transactions. |
| yum provides command | dnf provides command | The behavior of the dnf provides command is aligned to how it's documented; whereas, during the execution of the yum provides command, Yum applies certain, undocumented behavior. For example, if you run the yum provides sandbox command, Yum applies extra heuristics to interpret the sandbox part of the command, then it sequentially prepends entries from the PATH environment variable to the command to match a file that's provided by a package. DNF doesn't emulate this undocumented behavior. |
| --enableplugins option | Not available | This option isn't documented for DNF, as all plugins are enabled by default. |
| throttle and bandwidth options | throttle and bandwidth options | In DNF, for simultaneous downloads, the total downloading speed is now throttled. This support wasn't available in the Yum v3 tool, as downloaders ran in different processes. |

| Yum v3 Feature, Command, or Option | DNF Feature, Command, or Option | Notable Differences |
|---|---|---|
| installonlypkgs option | installonlypkgs | DNF appends the list values from the installonlypkgs configuration option to DNF defaults. Yum v3 overwrites the defaults by option values. |
| deltarpm_percentage option | Not available | The Boolean deltarpm option controls whether delta RPM files are used. Yum DNF doesn't support the use of the deltarpm_percentage option. Instead, the tool chooses the best value of the DRPM/RPM ratio to decide whether using deltarpm is appropriate in a particular situation. |
| .srpm files and non-existent package handling | .srpm files and non-existent package handling | DNF stops early with an error if a command requesting an installing operation on a local .srpm file is run. Yum v3 issues a warning and continues by installing the tour package. Note that Yum DNF emits the same error for package specifications that don't match any available package. |
| Promoting a package to install to a package that obsoletes it. | Promoting a package to install to a package that obsoletes it. | DNF doesn't automatically replace a request to install a package (A) by installing another package (B) if package B would obsolete package(A. The Yum v3 behavior is to perform the action if the obsoletes configuration option is enabled. However, note that this behavior isn't documented and can be harmful. |
| --installroot option | --installroot option | DNF provides more predictable behavior for this option and handles the path differently than the --config option, where this path is always related to the host system. Yum v3 combines this path with the installroot option. The reposdir option is also handled slightly differently in Yum DNF. For example, if one reposdirs path exists inside installroot, then repositories are taken only from installroot. Whereas, Yum v3 tests each path from reposdir. |

| Yum v3 Feature, Command, or Option | DNF Feature, Command, or Option | Notable Differences |
|---|---|---|
| Prompts displayed after a transaction table | Prompts displayed after a transaction table | The prompts that are displayed after a transaction table are different in DNF than they're for Yum v3. DNF doesn't provide download functionality after displaying the transaction table. You're only prompted to continue with the transaction or not. To download packages, use the download command. |
| list command | list command | The DNF behavior for this command is to list all packages from all repositories, which means there can be duplicate package names with different repository names listed. This change was made to enable users to choose a preferred repository. |

There isn't a direct replacement for the yum-updateonboot command in DNF. However, you can obtain a similar result by running the dnfautomatic command.

The following table compares Yum V3 plugins with DNF plugins.

| Yum Version 3 Plugin | DNF Plugin | Package |
|---|---|---|
| yum check | dnf repoquery --unsatisfied | dnf |
| yum-langpacks | | dnf-langpacks |
| yum-plugin-auto-update-debug-info | Option in debuginfo-install.conf | dnf-plugins-core |
| yum-plugin-copr | dnf copr | dnf-plugins-core |
| yum-plugin-fastestmirror | fastestmirror option in dnf.conf | dnf |
| yum-plugin-fs-snapshot | | dnf-plugins-extras-snapper |
| yum-plugin-local | | dnf-plugins-core |
| yum-plugin-merge-conf | | dnf-plugins-extras-rpmconf |
| yum-plugin-priorities | priority option in dnf.conf | dnf |
| yum-plugin-remove-with-leaves | dnfautoremove | dnf |
| yum-plugin-show-leaves | | dnf-plugins-core |
| yum-plugin-versionlock | | dnf-plugins-core |
| yum-rhn-plugin | | dnf-plugin-spacewalk |

The following table compares Yum v3 utilities with DNF plugins.

| Yum Version 3 Utility | DNF Plugin | DNF Package |
|---|---|---|
| debuginfo-install | dnf debuginfo-install | dnf-plugins-core |

| Yum Version 3 Utility | DNF Plugin | DNF Package |
|---|---|---|
| find-repos-of-install | dnf list installed | dnf |
| needs-restarting | dnf tracer | dnf-plugins-extras-tracer |
| package-cleanup | dnf list, dnf repoquery | dnf, dnf-plugins-core |
| repoclosure | dnf repoclosure | dnf-plugins-extras-repoclosure |
| repodiff | dnf repodiff | dnf-plugins-core |
| repo-graph | dnf repograph | dnf-plugins-extras-repograph |
| repomanage | dnf repomanage | dnf-plugins-extras-repomanage |
| repoquery | dnf repoquery | dnf |
| reposync | dnf reposync | dnf-plugins-core |
| repotrack | dnf download –resolve –alldeps | dnf-plugins-core |
| yum-builddep | dnf builddep | dnf-plugins-core |
| yum-config-manager | dnf config-manager | dnf-plugins-core |
| yum-debug-dump | dnf debug-dump | dnf-plugins-extras-debug |
| yum-debug-restore | dnf debug-restore | dnf-plugins-extras-debug |
| yumdownloader | dnf download | dnf-plugins-core |

The following table lists the Yum v3 package-cleanup command and its DNF replacement.

| Yum Version 3 Command | DNF Command |
|---|---|
| package-cleanup--dupes | dnfrepoquery--duplicates |
| package-cleanup--leaves | dnfrepoquery--unneeded |
| package-cleanup--orphans | dnfrepoquery--extras |
| package-cleanup--oldkernels | dnfrepoquery--installonly |
| package-cleanup--problems | dnfrepoquery--unsatisfied |
| package-cleanup--cleandupes | dnfremove--duplicates |
| package-cleanup--oldkernels | dnfremove--oldinstallonly |

A

# Application Stream Life Cycle

While the core operating system packages in the BaseOS repository for Oracle Linux 8, Oracle Linux 9, and Oracle Linux 10 retain a standard Oracle Linux support life cycle, the separate AppStream packages have their own major version releases and might have shorter lifespans from 2 to 5 years.

Support for the AppStream packages is limited to package installation help only. More support for AppStream packages might be provided if references to these packages and their use are included in other official Oracle Linux documentation from Oracle. Critical security errata and select high-impact critical bug fixes are provided in the newer versions of AppStream packages.

See Oracle Linux: Product Life Cycle Information for more information on the Application Stream Life Cycle durations for Oracle Linux releases.