



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

---

## Session - 32 Agenda:

### 1. Package Management Using Ansible

---

#### Package Management:

Most Linux distributions have a package management system with online repositories containing thousands of packages. This makes it very easy to install and remove applications, operating system components, documentation and much more.

S. No.	Linux Flavor	Package Format	Tools
1.	Debian (Ubuntu)	.deb	dpkg, apt-get & aptitude
2.	Fedora (RedHat)	.rpm	rpm, yum & dnf

#### Package Terminology:

##### Repository:

A Linux repository is a storage location that contains essential and popular software for different Linux distributions and, each distribution has its own official repositories (also called standard-repositories).

##### Dependency:

Some packages need other packages to function. Tools like apt-get, aptitude and yum will install all dependencies you need. When using dpkg or rpm, or when building from source, you will need to install dependencies yourself.

##### Open source:

These repositories contain a lot of independent open-source software. Often the source code is customized to integrate better with your distribution. Most distributions also offer this modified source code as a package in one or more source repositories.

##### DEB:

Most people use aptitude or apt-get to manage their Debian/Ubuntu family of Linux distributions. Both are a front end for dpkg and are themselves a back end for synaptic and other graphical tools.

The low-level tool to work with .deb packages is dpkg. Here you see how to obtain a list of all installed packages on a Debian server.

```
$ sudo dpkg -l
```

Here is an example on how to get information on an individual package.

```
$ sudo dpkg -l rsync
```

You can find the package that installed a certain file on your computer with dpkg -S.

```
$ sudo dpkg -S /usr/share/man/man5/passwd.5.gz
```

You can also get a list of all files that are installed by a certain program.

```
$ sudo dpkg -L bash
```

You could use dpkg -i to install a package and dpkg -r to remove a package, but you'd have to manually keep track of dependencies.

```
$ wget http://archive.ubuntu.com/ubuntu/pool/main/z/zsh/zsh\_5.8.1-1\_amd64.deb
```

```
$ ls -lh zsh*
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
$ sudo dpkg -i zsh*  
$ sudo dpkg -r zsh*  
$ sudo apt-get remove zsh
```

Debian has been using apt-get to manage packages since 1998. Today Debian and many Debian-based distributions still actively support apt-get, though some experts claim aptitude is better at handling dependencies than apt-get.

When typing apt-get update you are downloading the names, versions and short description of all packages available on all configured repositories for your system.

```
$ sudo apt-get update
```

Run apt-get update every time before performing other package operations.

One of the nicest features of apt-get is that it allows for a secure update of all software currently installed on your computer with just one command.

```
$ sudo apt-get upgrade
```

*In a nutshell, "apt-get update" updates the package index files, whereas "apt-get upgrade" upgrades the actual packages installed on your system.*

apt-get keeps a copy of downloaded packages in /var/cache/apt/archives, which can be removed using apt-get clean command.

```
$ ll /var/cache/apt/archives  
$ sudo apt-get clean
```

You can install one or more applications by appending their name behind apt-get install.

```
$ sudo apt-get install rsync
```

You can remove one or more applications by appending their name behind apt-get remove.

```
$ sudo apt-get remove rsync
```

Note however that some configuration information is not removed.

```
$ sudo dpkg -l rsync | tail -1 | tr -s ' '
```

You can purge one or more applications by appending their name behind apt-get purge.

```
$ sudo apt-get purge rsync  
$ sudo dpkg -l rsync | tail -1 | tr -s ' '
```

## **APTITUDE:**

Most people use aptitude for package management on Debian, Mint and Ubuntu systems.

```
$ sudo apt-get install aptitude -y
```

To synchronize with the repositories.

```
$ sudo aptitude update
```

To patch and upgrade all software to the latest version on Debian.

```
$ sudo aptitude upgrade
```

To patch and upgrade all software to the latest version on Ubuntu and Mint.



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
$ sudo aptitude safe-upgrade
```

To install an application with all dependencies.

```
$ sudo aptitude install zsh -y
```

To search the repositories for applications that contain a certain string in their name or description.

```
$ sudo aptitude search rsync
```

To remove an application.

```
$ sudo aptitude remove zsh -y
```

To remove an application and all configuration files.

```
$ sudo aptitude purge zsh -y
```

Both apt-get and aptitude use the same configuration information in /etc/apt/. Thus adding a repository for one of them, will automatically add it for both.

```
$ sudo cat /etc/apt/sources.list
```

## RPM:

The Red Hat package manager can be used on the command line with rpm or in a graphical way going to Applications--System Settings--Add/Remove Applications.

To obtain a list of all installed software, use the rpm -qa command.

```
# rpm -qa
```

```
# rpm -qa | grep chrony
```

To verify whether one package is installed, use rpm -q.

```
# rpm -q bash
```

To install or upgrade a package, use the -Uvh switches.

```
# wget http://mirror.centos.org/altarch/7/os/aarch64/Packages/zsh-5.0.2-34.el7_8.2.aarch64.rpm
```

```
# rpm -Uvh zsh-5.0.2-34.el7_8.2.aarch64.rpm
```

To remove a package, use the -e switch.

```
# rpm -e zsh
```

rpm -e verifies dependencies, and thus will prevent you from accidentally erasing packages that are needed by other packages.

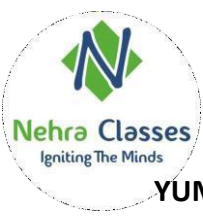
The rpm database is located at /var/lib/rpm. This database contains all meta information about packages that are installed (via rpm). It keeps track of all files, which enables complete removes of software.

```
# ls -lh /var/lib/rpm
```

To install/upgrade any package using rpm command we need to manually resolve the dependencies first.

```
# rpm -ivh zsh-5.0.2-34.el7_8.2.aarch64.rpm
```

If you want that system should automatically identify and resolve the dependencies at its own, you can use yum/dnf tools.



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

## **YUM:**

The Yellowdog Updater Modified (yum) is an easier command to work with rpm packages. It is installed by default on Fedora and Red Hat Enterprise Linux since version 5.2.

Issue yum list available to see a list of available packages.

```
# yum list
```

```
# yum list all
```

```
# yum list zsh
```

To search for a package containing a certain string in the description or name use yum search \$string.

```
# yum search zsh
```

To see whether a package is installed or not.

```
# yum list installed zsh
```

To search for a package containing a certain file.

```
# yum provides /usr/share/man/man5/passwd.5.gz
```

To install an application, use yum install \$package. Naturally yum will install all the necessary dependencies.

```
# yum install zsh -y
```

To bring all applications up to date, by downloading and installing them, issue yum update.

```
# yum update
```

If you only want to update one package, use yum update \$package.

```
# yum update sudo
```

To see the yum history.

```
# yum history
```

To downgrade a package, we can use yum downgrade \$package.

```
# yum downgrade sudo
```

To see the details of yum history event.

```
# yum history info 2
```

To undo the changes using yum history.

```
# yum history undo 2
```

Issue yum grouplist to see a list of all available software groups.

```
# yum grouplist
```

To install a set of applications, brought together via a group, use yum groupinstall \$groupname.

```
# yum groupinstall 'Server With GUI'
```

## **DNF:**

The DNF command (Dandified yum) is the next-generation version of the traditional YUM package manager for RedHat based systems. It is the default package manager for Fedora 22, CentOS8, and RHEL8. It is intended to be a replacement for YUM. It does Package Management



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

using RPM and libsolv (maintained by OpenSUSE).

Issue dnf list available to see a list of available packages.

```
# dnf list
```

```
# dnf list all
```

```
# dnf list zsh
```

To search for a package containing a certain string in the description or name use dnf search \$string.

```
# dnf search zsh
```

To search for a package containing a certain file.

```
# dnf provides /usr/share/man/man5/passwd.5.gz
```

To install an application, use yum install \$package. Naturally yum will install all the necessary dependencies.

```
# dnf install zsh
```

To bring all applications up to date, by downloading and installing them, issue yum update.

```
# dnf update
```

If you only want to update one package, use yum update \$package.

```
# dnf update sudo
```

To downgrade a package, we can use yum downgrade \$package.

```
# dnf downgrade sudo
```

To see the yum history.

```
# dnf history
```

To see the details of yum history event.

```
# dnf history info 2
```

To undo the changes using yum history.

```
# dnf history undo 2
```

Issue yum grouplist to see a list of all available software groups.

```
# dnf grouplist
```

To install a set of applications, brought together via a group, use yum groupinstall \$groupname.

```
# dnf groupinstall 'Server With GUI'
```

## YUM Repository:

The configuration of yum/dnf repositories is done in /etc/yum/yum.conf and /etc/yum/repos.d/ Configuring yum itself is done in /etc/yum.conf. This file will contain the location of a log file and a cache directory for yum and can also contain a list of repositories.

```
# cat /etc/yum.conf
```

Recently yum started accepting several repo files with each file containing a list of repositories. These repo files are located in the /etc/yum.repos.d/ directory.

```
# cd /etc/yum.repo.d/
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
# ls -lh *
```

You can easily enable/disable the repository by changing the parameter in the repo file or by issuing the commands.

you can use yum/dnf repolist command to see the status of available repositories.

```
# yum repolist all
```

## Package Management Using Ansible:

There are two ways of do the package management on the managed nodes using Ansible:

1. Ansible Ad-Hoc Commands
2. Ansible Playbooks

### 1. Package Management Using Ansible Ad-Hoc Commands:

We can issue command in form of Ansible Ad-hoc Commands to do the package management on the managed nodes.

```
$ ansible node1 -m command -a 'yum repolist'
```

```
$ ansible node1 -m yum_repository -a 'name=epel description=EPEL Repo baseurl=https://download.example/pub/epel/$releasever/Everything/$basearch/ metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-$releasever&arch=$basearch&infra=$infra&content=$contentdir enabled=true gpgcheck=true gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever' -b
```

```
$ ansible node1 -m command -a 'yum repolist'
```

```
$ ansible node1 -m command -a 'cat /etc/yum.repos.d/epel.repo'
```

```
$ ansible node1 -m yum_repository -a 'name=epel description=EPEL Repo baseurl=https://download.example/pub/epel/$releasever/Everything/$basearch/ metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-$releasever&arch=$basearch&infra=$infra&content=$contentdir enabled=false gpgcheck=true gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever' -b
```

```
$ ansible node1 -m command -a 'yum repolist'
```

```
$ ansible node1 -m command -a 'cat /etc/yum.repos.d/epel.repo'
```

```
$ ansible node1 -m yum_repository -a 'name=epel description=EPEL Repo baseurl=https://download.example/pub/epel/$releasever/Everything/$basearch/ metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-$releasever&arch=$basearch&infra=$infra&content=$contentdir enabled=false gpgcheck=true gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever state=absent' -b
```

```
$ ansible node1 -m command -a 'yum repolist'
```

```
$ ansible node1 -m command -a 'cat /etc/yum.repos.d/epel.repo'
```

```
$ ansible node1 -m command -a 'rpm -qa'
```

```
$ ansible node1 -m command -a 'rpm -qa | grep chrony'
```

```
$ ansible node1 -m command -a 'rpm -q bash'
```

```
$ ansible node1 -m command -a 'yum list'
```

```
$ ansible node1 -m command -a 'yum list zsh'
```

```
$ ansible node1 -m command -a 'yum search httpd'
```

```
$ ansible node1 -m command -a 'yum list installed chrony'
```

```
$ ansible node1 -m command -a 'yum install zsh -y' -b
```

```
$ ansible node1 -m command -a 'yum update -y' -b
```

```
$ ansible node1 -m command -a 'yum update sudo -y' -b
```

```
$ ansible node1 -m command -a 'yum history' -b
```

```
$ ansible node1 -m command -a 'dnf list zsh'
```

```
$ ansible node1 -m command -a 'dnf search httpd'
```

```
$ ansible node1 -m command -a 'dnf list installed chrony'
```

```
$ ansible node1 -m command -a 'dnf install zsh -y' -b
```

```
$ ansible node1 -m command -a 'dnf update -y' -b
```

```
$ ansible node1 -m command -a 'dnf update sudo -y' -b
```

```
$ ansible node1 -m command -a 'dnf history' -b
```

```
$ ansible node1 -m yum -a 'list=httpd'
```

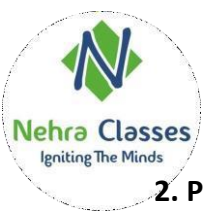
```
$ ansible node1 -m yum -a 'name=httpd state=latest' -b
```

```
$ ansible node1 -m yum -a 'name=httpd state=removed' -b
```

```
$ ansible node1 -m yum -a 'name=* state=latest' -b
```

```
$ ansible node1 -m package -a 'name=httpd state=latest' -b
```

```
$ ansible node1 -m package -a 'name=httpd state=absent' -b
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

## 2. Package Management Using Ansible Playbooks:

We can create ansible playbooks to do the package management on the managed nodes.

```
$ vim yum_repo.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Add repository
```

```
  yum_repository:
```

```
    name: epel
```

```
    description: EPEL YUM repo
```

```
    baseurl: https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
```

```
...
```

```
$ ansible-playbook yum_repo.yml
```

```
$ ansible node1 -m command -a 'yum repolist'
```

```
$ ansible node1 -m command -a 'ls /etc/yum.repos.d/'
```

```
$ ansible node1 -m command -a 'cat /etc/yum.repos.d/epel.repo'
```

```
$ vim yum_repo2.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Add multiple repositories into the same file (1/2)
```

```
  yum_repository:
```

```
    name: epel2
```

```
    description: EPEL YUM repo
```

```
    file: external_repos
```

```
    baseurl: https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
```

```
    gpgcheck: no
```

```
...
```

```
$ ansible-playbook yum_repo2.yml
```

```
$ ansible node1 -m command -a 'yum repolist'
```

```
$ ansible node1 -m command -a 'ls /etc/yum.repos.d/'
```

```
$ vim yum_repo3.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Remove repository
```

```
  yum_repository:
```

```
    name: epel
```

```
    state: absent
```

```
...
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
$ ansible-playbook yum_repo3.yml
$ ansible node1 -m command -a 'yum repolist'
$ ansible node1 -m command -a 'ls /etc/yum.repos.d/'
```

```
$ vim yum_repo4.yml
```

```
---
```

```
- name: Yum Repository Management
  hosts: node1
  become: true
  tasks:
    - name: Remove repository from a specific repo file
      yum_repository:
        name: epel2
        file: external_repos
        state: absent
```

```
...
```

```
$ ansible-playbook yum_repo4.yml
$ ansible node1 -m command -a 'yum repolist'
$ ansible node1 -m command -a 'ls /etc/yum.repos.d/'
```

```
$ vim yum.yml
```

```
---
```

```
- name: Yum Repository Management
  hosts: node1
  become: true
  tasks:
    - name: Install the latest version of Apache
      yum:
        name: httpd
        state: latest
```

```
...
```

```
$ ansible-playbook yum.yml
$ ansible node1 -m command -a 'rpm -q httpd'
```

```
$ vim yum2.yml
```

```
---
```

```
- name: Yum Repository Management
  hosts: node1
  become: true
  tasks:
    - name: Install a list of packages.
      yum:
        name:
          - nginx
          - postgresql
          - postgresql-server
        state: present
```

```
...
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
$ ansible-playbook yum2.yml
```

```
$ ansible node1 -m command -a 'rpm -q nginx'
```

```
$ vim yum3.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Install a list of packages with a list variable
```

```
  yum:
```

```
    name: "{{ packages }}"
```

```
  vars:
```

```
    packages:
```

```
      - httpd
```

```
      - httpd-tools
```

```
      - ftp
```

```
      - vsftpd
```

```
...
```

```
$ ansible-playbook yum3.yml
```

```
$ ansible node1 -m command -a 'rpm -q ftp'
```

```
$ vim yum4.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Install the latest version of Apache
```

```
  yum:
```

```
    name: httpd
```

```
    state: absent
```

```
...
```

```
$ ansible-playbook yum4.yml
```

```
$ ansible node1 -m command -a 'rpm -q httpd'
```

```
$ vim yum5.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Upgrade all packages
```

```
  yum:
```

```
    name: '*'
```

```
    state: latest
```

```
...
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
$ ansible-playbook yum5.yml
$ ansible node1 -m command -a 'yum history'
```

```
$ vim yum6.yml
```

```
---
```

```
- name: Yum Repository Management
hosts: node1
become: true
tasks:
  - name: Upgrade all packages, excluding kernel & foo related packages
    yum:
      name: '*'
      state: latest
      exclude: kernel*,chrony*
```

```
...
```

```
$ ansible-playbook yum6.yml
$ ansible node1 -m command -a 'yum history'
```

```
$ vim yum7.yml
```

```
---
```

```
- name: Yum Repository Management
hosts: node1
become: true
tasks:
  - name: Install the nginx rpm from a remote repo
    yum:
      name: http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6ngx.noarch.rpm
      state: present
      disable_gpg_check: true
```

```
...
```

```
$ ansible-playbook yum7.yml
```

```
$ ansible node1 -m command -a 'wget http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6ngx.noarch.rpm'
```

```
$ ansible node1 -m command -a 'ls /home/vikasnehra/'
```

```
$ vim yum8.yml
```

```
---
```

```
- name: Yum Repository Management
hosts: node1
become: true
tasks:
  - name: Install nginx rpm from a local file
    yum:
      name: /home/vikasnehra/nginx-release-centos-6-0.el6ngx.noarch.rpm
      state: present
      disable_gpg_check: true
```

```
...
```

```
$ ansible-playbook yum8.yml
```



# Package Management Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
$ vim yum9.yml
```

```
---
```

```
- name: Yum Repository Management
```

```
hosts: node1
```

```
become: true
```

```
tasks:
```

```
- name: Install the 'Development tools' package group
```

```
  yum:
```

```
    name: "@Development tools"
```

```
    state: present
```

```
...
```

```
$ ansible-playbook yum9.yml
```

---

Thank You

**Nehra Classes**  
Igniting The Minds