Ansible Yum

Ansible has a specific module for managing the Yum packages. You can install, remove, upgrade or downgrade versions and many more by using this module.

The Yum module also requires two parameters for the primary command, like other package management modules in Ansible.

- Name: provides the name of the package which you want to install.
- State: maintains the state of the packages, like what should be the state of the package after the task is completed (present or absent). By default, the value of the parameter is "present".

Installing a Package

Let's install the git package using the Yum module. Set the name parameter to "git" and the state parameter to "present".

```
- hosts: all
  tasks:
  - name: Install yum package in Ansible example
    yum:
      name: git
      state: present
```

If the package was not on the remote server, then the latest version will be installed.

And if the package was already installed on the remote server, then it will not be updated to a new version because the "state" is already "present".

NOTE: Both the "present" and "installed" parameters have the same behavior.

Installing the latest Version

If you want to install the newest version, then you can set the state parameter to "latest". It will install the newest package, whether the package is present or not.

```
- hosts: all
```

```
  tasks:
  - name: Install the latest yum package example.
    yum:
      name: git
      state: latest
```

Installing a Specific Version

Sometimes you want to install a particular version of the packages. You can do this by appending the version with the package name.

```
<packagename>-<package version>
```

For example git-1.8.3.1-6.el7

Let's install the git package with version and release, 1.8.3.1-6.el7, on the remote server.

```
- hosts: all
  tasks:
  - name: Install a specific version of a package in Ansible.
    yum:
      name: git-1.8.3.1-6.el7
      state: present
```

Output

```
------
[root@rpm ~]# yum info git
Installed Packages
Name: git
Arch: x86_64
Version: 1.8.3.1
Release: 6.el7
```

Installing Multiple Packages

If you want to install various packages, you can do this by using the "with_items" statement to loop through a list of the packages.

Let's execute three Yum packages such as MySQL, git, and httpd.

```
- hosts: all
  tasks:
  - name: yum
    yum:
      name: "{{ item }}"
      state: present
    with_items:
    - git
    - httpd
    - mysql
```

NOTE: The above code will not be executed as a single package installation in each loop instance. Instead, all the modules are installed in one go. This optimized form is the behavior since 1.9.2.

Update all Packages

You can update all the yum packages, like giving the command yum -y update. You can use the wildcard "*" in the name.

```
- hosts: all
  tasks:
  - name: Upgrade all yum packages ansible.
    yum:
      name: "*"
      state: latest
```

Also, you can use the exclude parameter so that some packages should not be upgraded. The following task will not update the git package.

```
- hosts: all
  tasks:
  - name: Exclude some packages from an upgrade in Ansible.
    yum:
      name: "*"
      state: latest
      exclude: git*
```