Contents

1 System Packages for Enterprise Linux Hosts

This role installs packages on Enterprise Linux (RHEL, CentOS, Fedora) hosts, including a default set of useful packages for sysadmins working on those hosts.

1.1 Bookkeeping and Usage

1.1.1 Requirements

This role uses only built-in Ansible modules, so no additional modules should be required.

1.1.2 Dependencies

No other roles are required to use this role.

1.1.3 Installing This Role

In order to install this role, you should create (or edit) a file called `requirements.yml` in your Ansible root with contents like this:

```
- src: https://github.com/jhenahan/system-packages-org
  version: v1.0.0 # corresponds to a git tag
```

You can now run `ansible-galaxy install -r requirements.yml` to install the role.

If you wish to use this role in a Tower job, add the `requirements.yml` file to the `roles` directory in your project and Tower will download the roles when the job runs.

1.1.4 Example Playbook

Assuming you have defined a host group called `servers`, you can use this role on the hosts in that inventory as follows.

```
- hosts: servers
  roles:
    - system-packages-org
```

1.1.5 License

1.1.6 Author Information

Jack Henahan

1.2 API

1.2.1 Variables

In general, you MAY override default variables, and you SHOULD NOT override role variables unless you have a good reason and know what you are doing. Default variables can be overridden anywhere (likely in `group_vars` or `host_vars`), but role variables can only be overridden according to the Ansible 2.x precedence rules. Configurations which use this role and override the role variables are considered unsupported.

1. Default Variables Because Enterprise Linux default repos do not always provide everything we want (or what they provide is old), we can enable additional repos to get access to a wider variety of packages. To disable them, set the following variable to `no`.

   ```
   enable_additional_repos: yes
   ```

   Additional repositories are specified as a list of dictionaries. Refer to the Ansible documentation for the `yum` repository module for field explanations. We only explicitly use `name`, `description`, `baseurl`, `metalink`, `gpgkey`, so review the requirements for those fields in the documentation. To specify the repository whitelist, include the key `whitelist` in your repository dictionary, with a YAML list of packages.

   ```
   additional_repos: []
   ```

You may list any packages your application (etc.) may require using the following variable.

```
packages: []
```

There's an additional package variable for adding additional packages at the host level (i.e., in `host_vars` or an inventory file).

```
host_packages: []
```

Finally, you can set the desired state of your package set. `latest`, the default, indicates that you wish to install the latest version available, and will additionally install updates as necessary to realize this condition. `present` will merely ensure that the packages are installed, with no automatic updates. If your host's package repos are managed by Satellite and your EPEL package versions are not sensitive, you can leave this as the default. Otherwise, you may wish to set it to `present` to prevent upgrades.

```
package_state: "latest"
```

2. Role Variables The default package set provides a variety of diagnostic tools, necessary libraries, and additional development and system administration tools.

```
default_packages:
  # shells
  - bash
  - bash-completion
  # editors
  - vim
  - vim-enhanced
  # package management
  - yum-utils.noarch
  # SELinux deps
  - policycoreutils-python
  # diagnostics
  - dstat
  - iptraf
  - lsof
  - lsscsi
  - psmisc
  - strace
  - sysstat
  # networking
  - bind-utils
  - net-snmp-utils
  - net-tools
  - samba-common
  - samba-winbind-modules
  - wget
  # authentication
  - krb5-workstation.x86_64
  - pam_krb5
  # build tools
  - autoconf
  # scripting
  - expect
  - perl
  - perl-core
  - python
  # tools
  - bzip2
  - git
  - rsync
  - screen
```

The EPEL package set is designed not to conflict with base distribution package sets, but has historically caused some issues when used in combination with other "additional packages" repos like RHEL Extras. For this reason, you must explicitly whitelist packages from EPEL to avoid a descent into madness.

By default, we enable EPEL with a reasonable whitelist, with some additional helpful tools and an alternative shell (`fish`).

```
epel_whitelist:
  - htop
  - iftop
  - multitail
  - fish
```

```
default_additional_repos:
  - name: epel
    description: "EPEL YUM repo"
    baseurl:
     ↪  "https://download.fedoraproject.org/pub/epel/7/$basearch"
    metalink:
     ↪  "https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch"
    gpgkey:
     ↪  "http://download.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-7"
    whitelist: "{{ epel_whitelist }}"
```

If you do not want EPEL in your additional repos, you can pass a redefinition of `default_additional_repos` as a role param like so:

```
- hosts: servers
  roles:
    - { role: system-packages-org,
     ↪  default_additional_repos: [] }
```

Similarly, if you do want to use EPEL, but you'd rather whitelist a different set of packages, you can pass `epel_whitelist` as a role param in the same way as above.

Finally, two helper variables are declared to merge the default and user-defined package sets and whitelists. Duplicate packages are automatically removed.

```
final_additional_repos: "{{ default_additional_repos |
 ↪  union(additional_repos) }}"
final_packages: "{{ default_packages | union(packages) }}"
```

1.3 Idempotence

In order to avoid cleaning the `yum` cache unnecessarily on future runs, we touch a file to mark our hosts as managed after our first run.

```
- name: "Mark host packages as managed"
  file:
    path: "/etc/yum/ansible-managed"
    state: touch
```

1.4 Tasks

1.4.1 Consistency

To ensure a consistent deployment for green and brown field deployments, we clean the `yum` cache once and mark the system as managed for future run.

```
- name: "[consistency] | Determine if we've managed this host
↪  before"
  stat:
    path: "/etc/yum/ansible-managed"
  register: "yum_managed"

- name: "[consistency] | Clean yum cache"
  command: "yum clean all"
  when: "not yum_managed.stat.exists"
  notify: "Mark host packages as managed"
```

1.4.2 Repos

Now we resolve any repository changes specified in the config. Disabled (i.e., `enable: false`) repos will be removed.

```
- name: "[repos] | Enable additional repos"
  yum_repository:
    name: "{{ item.name }}"
    description: "{{ item.description | default(omit) }}"
    baseurl: "{{ item.baseurl | default(omit) }}"
    metalink: "{{ item.metalink | default(omit) }}"
    gpgkey: "{{ item.gpgkey | default(omit) }}"
    includepkgs: "{{ item.whitelist | default(None) | join(' ')
    ↪  or omit }}"
    state: "{{ item.enable | default(enable_additional_repos) |
    ↪  ternary('present', 'absent') }}"
  loop: "{{ final_additional_repos }}"
```

1.4.3 Packages

For sanity's sake, we first resolve whatever additional repos have been changed. This means installing/updating packages from any active repo, and removing packages known to be from those repos. This needs some work to really work automatically in a brown field environment (since we don't know what, if anything, was whitelisted then), but for ongoing management of hosts it should be sufficient. It also can't handle removing packages that are removed from a whitelist between runs, but that's outside the scope of this role.

Because we want to handle an arbitrary number of additional repos, we fob the work off on `include_tasks`.

```
- name: "[packages] | Include package resolution tasks"
  include_tasks: resolve-additional-repos.yml
  loop: "{{ final_additional_repos }}"
```

While processing each repo, we capture its activation state so we can correctly add/remove packages. We inherit the package state from the settings already discussed.

```
- name: "[additional-repo] | Get repo activation state for {{
↪  item.name }}"
  set_fact:
    additional_repo_enabled: "{{ item.enabled |
    ↪  default(enable_additional_repos) }}"

- name: "[additional-repo] | Resolve additional repo packages
↪  for {{ item.name }}"
  yum:
    state: "{{ additional_repo_enabled | ternary(package_state,
    ↪  'absent') }}"
    name: "{{ package }}"
  loop: "{{ item.whitelist }}"
  loop_control:
    loop_var: package
```

Finally, we install/update our base package set.

```
- name: "[packages] | Install system packages"
  yum:
    state: "{{ package_state }}"
    name: "{{ item }}"
  loop: "{{ final_packages }}"
```

1.5 Ansible Galaxy Meta Information

```
galaxy_info:
  author: Jack Henahan
  description: Baseline package sets for Enterprise Linux hosts
  issue_tracker_url:
   ↪  https://github.com/jhenahan/system-packages-org/issues
  license: BSD
  min_ansible_version: 2.5
  platforms:
  - name: EL
    versions:
    - 7
  galaxy_tags:
    - baseline
    - redhat
    - centos

dependencies: []
```