Patch reporting

When did I do what?

Jan Bundesmann

THE Linux & Open Source Company!

Consulting

Engineering

Support

Training

AlmaLinux · amazon web services · ANSIBLE · Azure · CentOS · debian · docker · HIPERWS · git · Google Cloud Partner

Istio · kafka · KAFKA · kubernetes · KVM · ORACLE Linux · oVirt · PROXMOX · puppet · RANCHER

Red Hat · TRAVIS CI · Rocky Linux · SALTSTACK · SUSE · Terraform · Ubuntu · vmware · Windows

Problem

What's the question?

- Some people want to know when they patched which package on which host.
- They expect foreman to have this information.

What's the problem?

Foreman / katello only knows the current package status.

Agenda

1. My solution
   - Storing information externally
   - All currently installed packages
2. Improvements

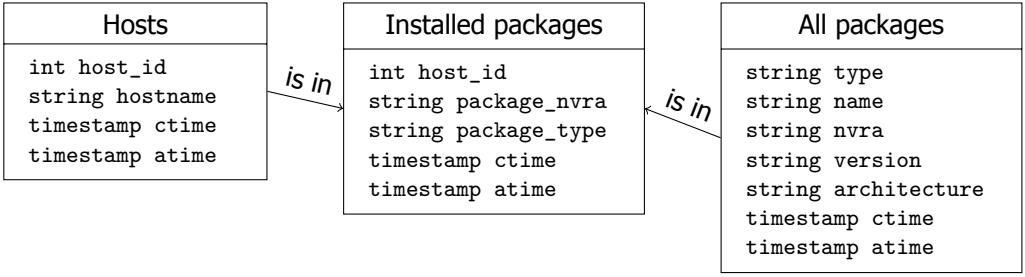
Agenda

1. My solution
   - Storing information externally
   - All currently installed packages
2. Improvements

Agenda

1. My solution
   - Storing information externally
   - All currently installed packages
2. Improvements

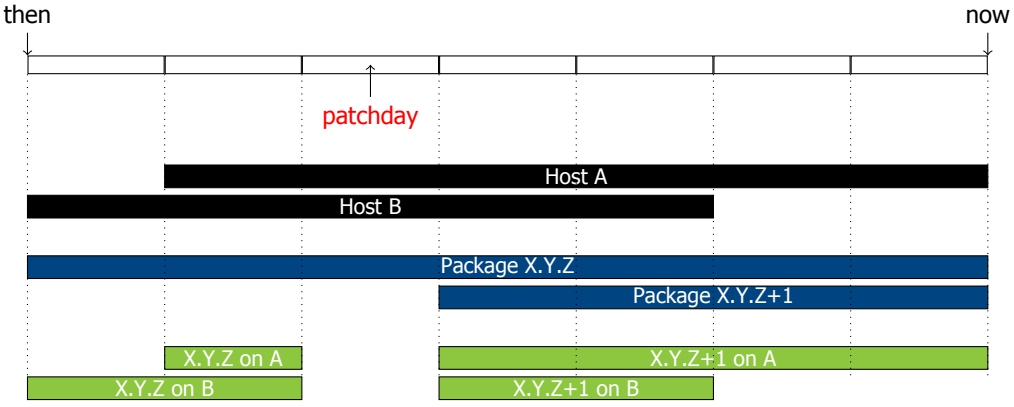
Storing information externally



How to work with this structure

- Postgres database with 3 tables
- Content updated regularly
- Dynamics stored in `ctime` and `atime`
  - `ctime`: entity appeared the first time
  - `atime`: entity appeared the last time

Visualize dynamics



Question that can be answered

- Which packages have been updated since the last patch day?
- Which packages have been updated within a given time slot?
- When has a package been update on host XY?
- Has a security issue on host XY been fixed?
- Which packages will be updated on the next patch day / upgrade slot?
- When did the last update on host XY happen?

Example query - packages newly installed after patchday

```
SELECT hosts.hostname, pkg_info_1.name, pkg_info_1.version as current_version

FROM hosts

JOIN installed_packages AS pkg_currently_installed ON (hosts.hostname = pkg_currently_installed.hostname)
JOIN all_packages AS pkg_info_1 ON (
    pkg_info_1.nvra = pkg_currently_installed.package_nvra and
    pkg_info_1.type = pkg_currently_installed.package_type
)

JOIN installed_packages AS pkg_ever_installed ON (hosts.hostname = pkg_ever_installed.hostname)
JOIN all_packages AS pkg_info_2 ON (
    pkg_info_2.nvra = pkg_ever_installed.package_nvra and
    pkg_info_2.type = pkg_ever_installed.package_type and
    pkg_info_1.name = pkg_info_2.name
)

WHERE hosts.atime >= '{{ recent }}'                              /* host still exists */
AND   hosts.ctime <= '{{ last_patchday }}'                       /* host already existed */
AND   pkg_currently_installed.ctime >= '{{ last_patchday }}'     /* installed after last patchday */

GROUP BY (
hosts.hostname,
pkg_info_1.name,
pkg_info_1.version
)
HAVING COUNT(*) = 1

ORDER BY hosts.hostname,pkg_info_1.name,pkg_info_1.version;
```

Agenda

1. My solution
   - Storing information externally
   - All currently installed packages
2. Improvements

First attempt: API

1. Query list of hosts
2. Query list of available packages
3. Query list installed on hosts

Second attempt: Report templates

```
<%#
name: All Hosts - Package Collector
model: ReportTemplate
%>
<%  load_hosts(search: '*').each_record do |host| -%>
<%    host.installed_packages.each do |pkg| -%>
<%
  architecture = pkg.nvra.gsub("#{pkg.name}-", "").match(/(?<architecture>[^.]*)$/)
  version = pkg.nvra.gsub("#{pkg.name}-", "").gsub(/\.#{architecture}/, "")
-%>
<%-   report_row(
        'hostid': host.id, 'hostname': host.name, 'packagetype': 'rpm',
        'packagename': pkg.name, 'packagenvra': pkg.nvra,
        'packageversion': version, 'packagearchitecture': architecture
      )-%>
<%    end -%>
<%    host.installed_debs.each do |pkg| -%>
<%-   report_row(
        'hostid': host.id, 'hostname': host.name, 'packagetype': 'deb',
        'packagename': pkg.name,
        'packagenvra': "#{pkg.name}-#{pkg.version}.#{pkg.architecture}",
        'packageversion': pkg.version, 'packagearchitecture': pkg.architecture
      )-%>
<%    end -%>
<%  end -%>
<%= report_render -%>
```

Problems and success

- Report templates much faster than direct API calls
- Doesn’t work with safemode rendering enabled, currently
- Unify format for RPM and DEB content
- Works on my system, doesn’t work for large number of hosts

Some nice implementation tweaks

- Streamline setup
  - Provision separate collector host using Ansible role, template, Application Centric Deployment
  - Define a single report template (in Ansible?)
- Easy usage
  - Playbook to collect data
  - Playbook to generate report

Agenda

1. My solution
   - Storing information externally
   - All currently installed packages
2. Improvements

Possible improvements

- Limit query to Host Collections or Host Groups
- Query PostgreSQL directly
- Time series database
- Metrics endpoint

Thank you for your attention!

Possible improvements

- Limit query to Host Collections or Host Groups
- Query PostgreSQL directly
- Time series database
- Metrics endpoint

Thank you for your attention!