

Journey of Automation - Github, Galaxy, Fedora

Pavel Cahyna, Sergei Petrosian

Automation is more important than ever before in software project management

Learn how the Linux System Roles([url](#)) team leverages automation:

1. Automated Ansible role release and publish to Ansible Galaxy
2. Automated Fedora RPM build and publish with Packit

Automated GitHub Releases

1. A script executed manually to create a new tag in GitHub repository
 - a. Identify a new semantic version
 - b. Generate changelog using conventional commits
 - c. Create a PR with updated changelog
2. Once the changelog PR is merged, a GitHub workflow triggers and does two tasks:
 - a. Tags and releases GitHub repository
 - b. Publishes repository content to Ansible Galaxy

Changelog Generation: Conventional Commits Format

Format:

```
<type><!>: PR Title
```

Example:

feat: Support custom data and logs storage paths

Figure out the new semantic version

- **!** - **MAJOR** bump
- **feat** - **MINOR** bump
- **fix**, **ci**, **test**,... - **PATCH** bump

For example:

feat: Add support for LVM RAID stripe size (#357) - **MINOR** bump

feat!: User-specified mount point owner and permissions (#239) - **MAJOR** bump

test: Add basic selinux_restore_dirs test - **PATCH** bump

Build changelog based on PR types

feat: -> **New Features**

fix: -> **Bug Fixes**

else -> **Other Changes**

GitHub Release Process using Conventional PR Titles

1. A developer runs the script
2. This script collect merged PRs since last release
3. Using conventional commits format, identifies version and generates a new changelog
4. Pushes a PR with the updated changelog

Creating Releases

PR merge triggers workflow that creates GitHub tag and release, and publishes repository content to Ansible Galaxy

```
name: Tag, release, and publish role based on CHANGELOG.md push
on:
  push:
    branches:
      - main
    paths:
      - CHANGELOG.md
jobs:
  - name: Create tag...
  - name: Create Release...
  - name: Publish role to Galaxy...
```


Automated RPM Release with Packit

[Packit Service](#) proposes Fedora releases from GitHub releases

- triggered by GitHub release
- updates spec file (`Version`, `%changelog`)
- uploads Sources to lookaside
- opens Pagure PR with updates
- performs Koji build & Bodhi update after Pagure PR is merged

Enabling Packit

Upstream (GitHub repo): create `.packit.yaml`

```
jobs:
- job: propose_downstream
  trigger: release
  dist_git_branches:
    - fedora-all
```

Downstream (Fedora dist-git): create `.packit.yaml`

```
jobs:
- job: koji_build
  trigger: commit
  dist_git_branches:
    - fedora-all
- job: bodhi_update
  trigger: commit
  dist_git_branches:
    - fedora-branched # rawhide updates are created automatically
```

Caveats (1)

Where to maintain the spec file?

If in GitHub repo, any Fedora changes will get overwritten.

Solution:

- keep spec file in Fedora dist-git repo
 - fetch it from there before creating the update

```
actions:  
  post-upstream-clone:  
    - "wget https://src.fedoraproject.org/rpms/linux-system-roles/raw/rawhide/f/linux-system-roles.spec -O linux-system-roles.spec"
```

together with all the files that it `%include`s:

```
- "wget https://src.fedoraproject.org/rpms/linux-system-roles/raw/rawhide/f/extrasources.inc -O extrasources.inc"
```

- or, configure Packit in Fedora dist-git instead of in GitHub repo
 - use `job: pull_from_downstream` instead of `job: propose_downstream`

Caveats (2)

How about RPM %changelog?

- by default, all Git commit message summaries in the GitHub repo used as the %changelog entry
- `copy_upstream_release_description` uses the GitHub release description.

Contrary to [Fedora packaging guidelines](#):

"They must never simply contain an entire copy of the source CHANGELOG entries."

Solution: custom changelog generator

```
actions:
  changelog-entry:
    - echo "- Rebase to version ${PACKIT_PROJECT_VERSION}"
```

Caveats (3)

Multi-Source RPMs

Our linux-system-roles RPM assembled from many individual repositories

Need: multiple Source: tags in spec, update them if any source tarball changes, Packit to upload them to lookaside

- using a spec file generator to update Sources

```
actions:
  post-upstream-clone:
    - "/generate_spec_sources.py linux-system-roles.spec.in linux-system-roles.spec"
```

```
# BEGIN AUTOGENERATED SOURCES
...
Source24: %{archiveurl24}
# END AUTOGENERATED SOURCES
```

- Packit now supports multi-source RPMs. If a Source is an URL, Packit uploads it to lookaside <https://github.com/packit/packit/pull/1778>