#CentOSClassroom at #SCaLE21x

centos@fosstodon.org CentOSProject centos-project CentOS

Integration SIG

ISA SIG

Automotive SIG

Docs SIG

Promo SIG

Storage SIG

NFV SIG

Artwork SIG

CentOS Stream

Hyperscale SIG

Cloud SIG

Kmods SIG

Infrastructure SIG

Virtualization SIG

Messaging SIG

Ops Tools SIG

Alt Images SIG

EPEL

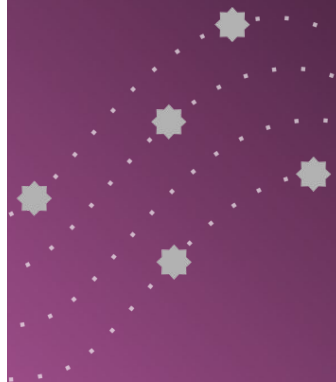
CentOS Stream

CentOS Stream

- Long-lifecycle distribution derived from Fedora Linux
- New major version released every 3 years
- Each major version is maintained for about 5.5 years
- Reference implementation of “Enterprise Linux”
- RHEL minor versions are derived from CentOS Stream major versions
- RHEL-style updates (i.e. preference for backports over rebases)

CentOS Stream

- Not a rolling release
- Not bleeding edge
- Not experimental



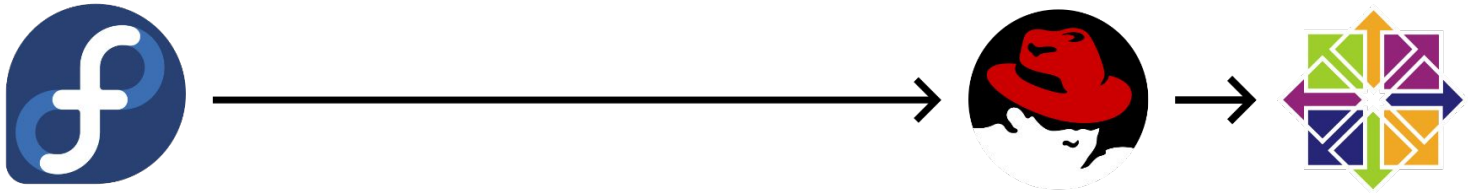
CentOS Stream

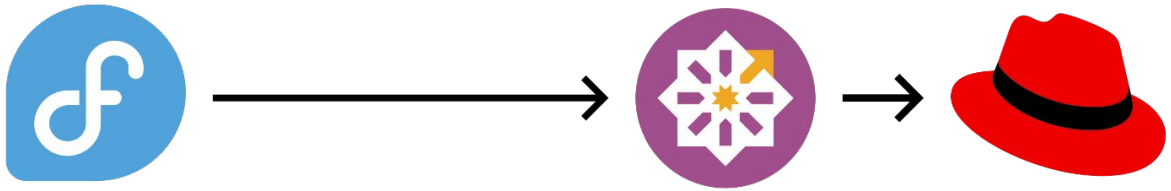
- RHEL maintainers are now also CentOS maintainers
- Often the Fedora maintainers as well
- Often involved in the upstream software project as well
- Holistic management across the ecosystem

CentOS Stream

- Fixes long standing problems with being a “clone”
- Can accept contributions from the community that change the operating system
- Can actually fix bugs instead of closing them outright in pursuit of being “bug-for-bug compatibility”

CentOS Stream

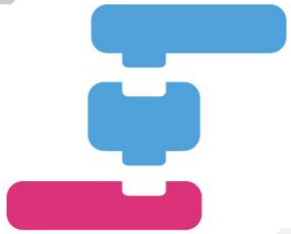




CentOS Stream

Fedora Linux integrates the latest and greatest software from across the open source ecosystem into an operating system.

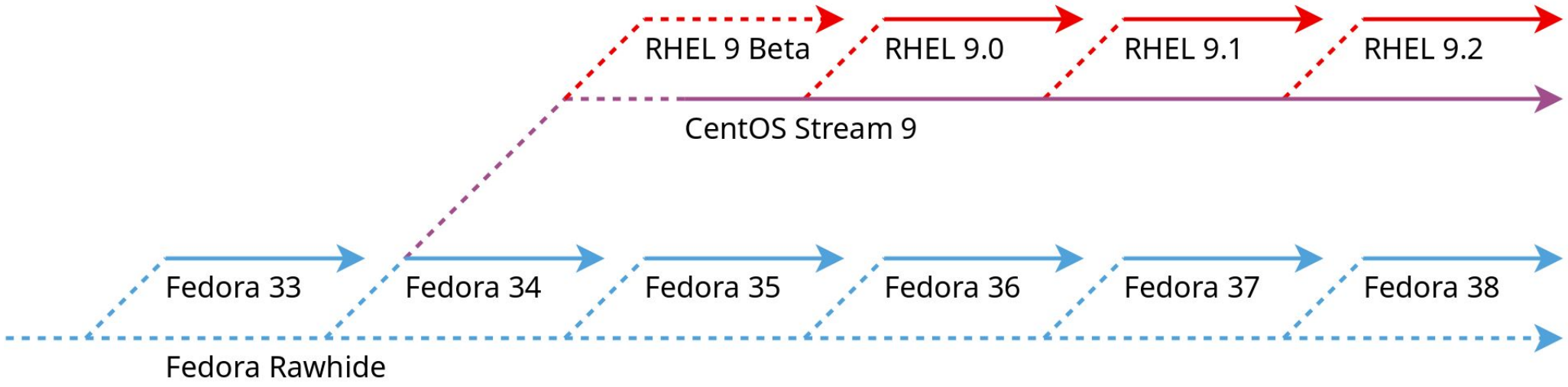
CentOS Stream is an operating system with a selection of packages for enterprise environments with continuous updates.



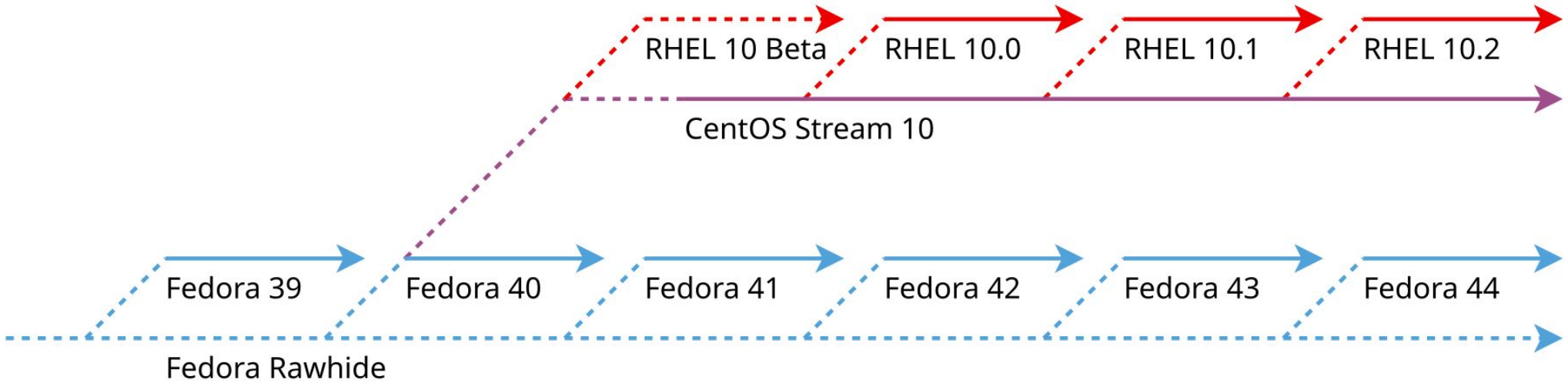
EPEL builds various additional packages from Fedora for CentOS and RHEL.

Red Hat Enterprise Linux is a production-grade operating system that provides a more secure, supported, and flexible foundation for critical workloads and applications.

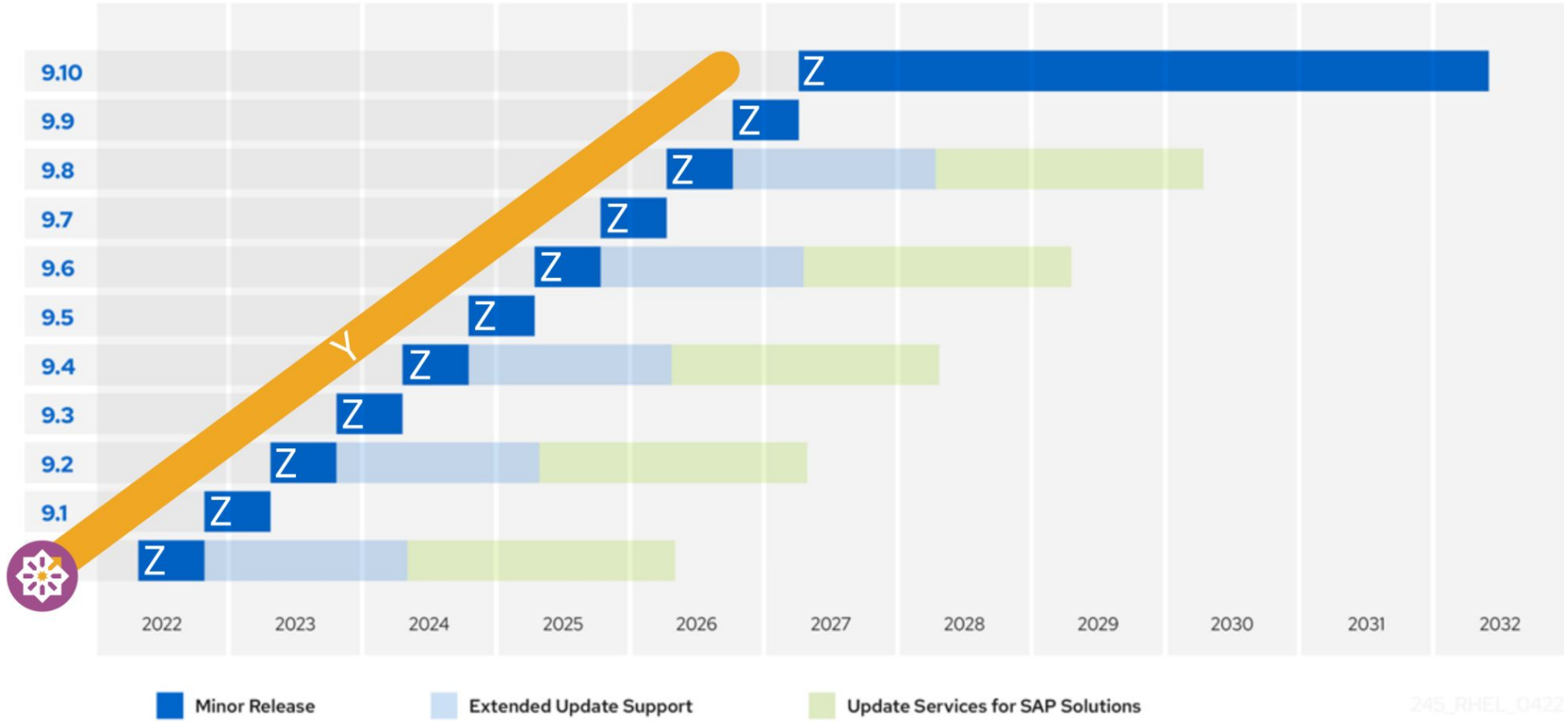
CentOS Stream



CentOS Stream



CentOS Stream



CentOS Board

CentOS Board

- Sets strategic direction for CentOS project
- Supposed to be non-technical (but…)
- 10 appointed members serve as long as they're active
- All but 1 serve as individuals, not as representatives
- Consensus model, nobody has magic veto

SIG Council (proposed)

SIG Council proposal

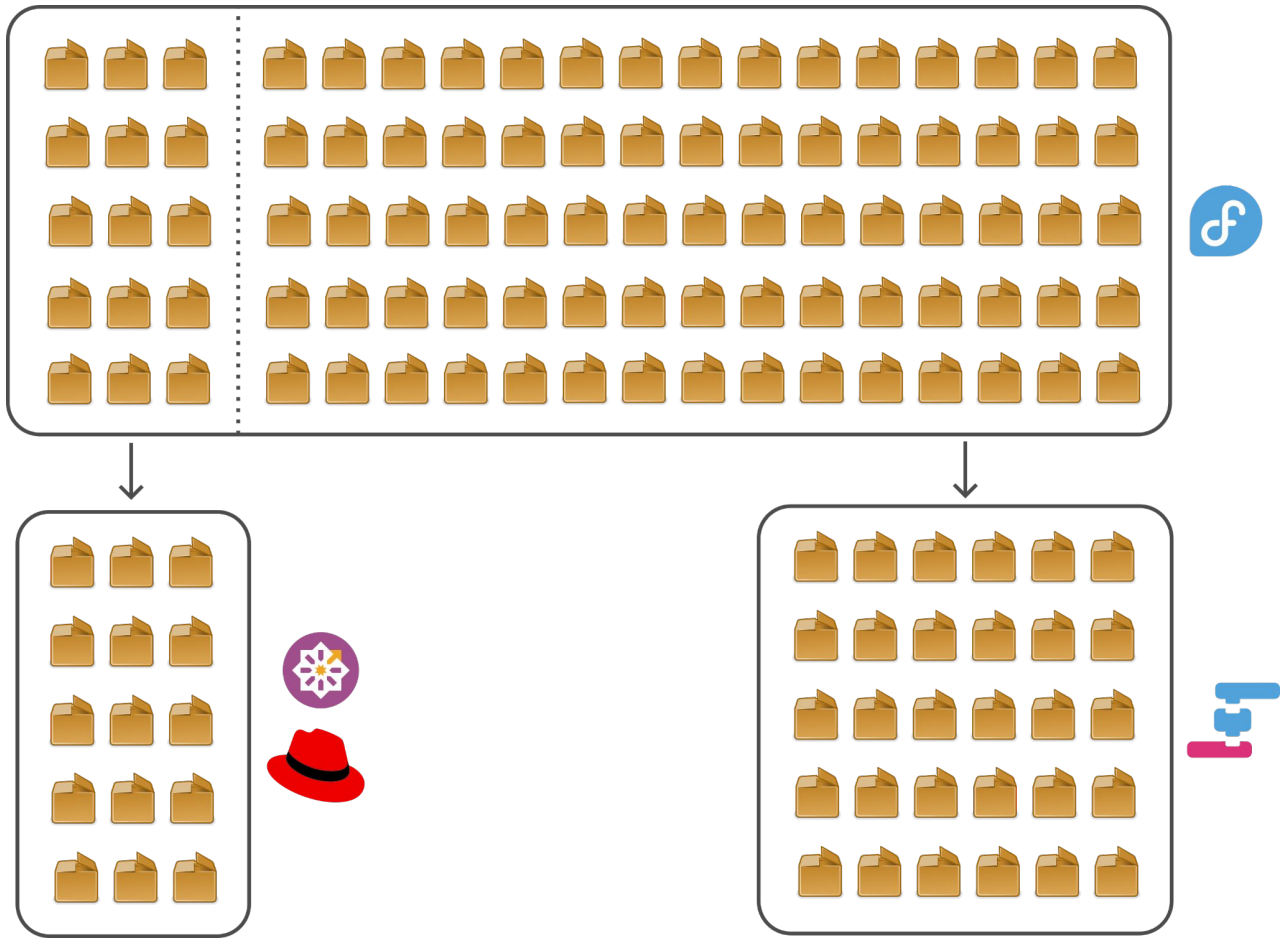
- Board is involved in too many technical discussions
- Need a voice for SIGs to talk to Stream and Infra
- And a way for SIGs to share with each other
- Includes SIG chairs, Stream Eng, and RHEL ENG
- Still a proposal (Board issue #126)

EPEL

EPEL

- Extra Packages for Enterprise Linux
- Part of Fedora Project
- Fedora packages, built to target CentOS Stream and RHEL

EPEL



Special Interest Groups (SIGs)

Integration SIG

Integration is verifying that products and services built on top of RHEL or CentOS Stream will continue to work on CentOS Stream and the next release of RHEL and will not break on package updates.

Integration SIG

Accomplishments

- Documented CentOS Stream gating process
- t_functional test suite migration to TMT framework
- Webhooks to relay CentOS events from GitLab to Fedora Message Bus

Work in Progress

- Git interface to the Compose Gate
- Continue the migration and enhancement of the compose tests
- Continue expanding the SIG Documentation

Hyperscale SIG

The Hyperscale SIG focuses on enabling CentOS Stream deployment on large-scale infrastructures and facilitating collaboration on packages and tooling.

Hyperscale SIG

SIG Content

- RPM Copy on Write
- latest systemd (255)
- Fedora-based kernel
- workstation spin
- container images
- cloud images (WIP)
- expanded virtualization stack (WIP)

CentOS Stream changes

- PipeWire and WirePlumber for audio
- JACK support in PipeWire
- systemd-oomd

Kmods SIG

The Kmods SIG focuses on packaging and maintaining kernel modules for CentOS Stream and Enterprise Linux.

- Builds kernel modules for things like filesystems and hardware controllers that aren't in CentOS Stream
- Builds modules for both CentOS Stream and RHEL.

Alternative Images SIG

The Alternate Images SIG's goal will be to build and provide alternate iso images for CentOS Stream.

Alternative Images SIG

- Worked with Koji to add and support integration for kiwi
- Worked with CentOS Infra to get the kiwi support deployed in CBS
- Built our first GNOME and KDE Plasma live ISOs for AArch64 and x86_64 and released them to users: https://mirror.stream.centos.org/SIGs/9-stream/altimages/images/live/
- Working on creating "minimal" Weston live ISO and "maximum" multi-desktop live ISO

ISA SIG

The purpose of the ISA SIG is to quantify the potential benefits of applying existing compiler technology to distribution packages, targeting more recent CPUs, and evaluating different options for how these optimizations can be maintained in a scalable way, and delivered to end users.

Automotive SIG

Public space for collaboration between third parties interested in open development of software targeted at in-vehicle automotive use cases.

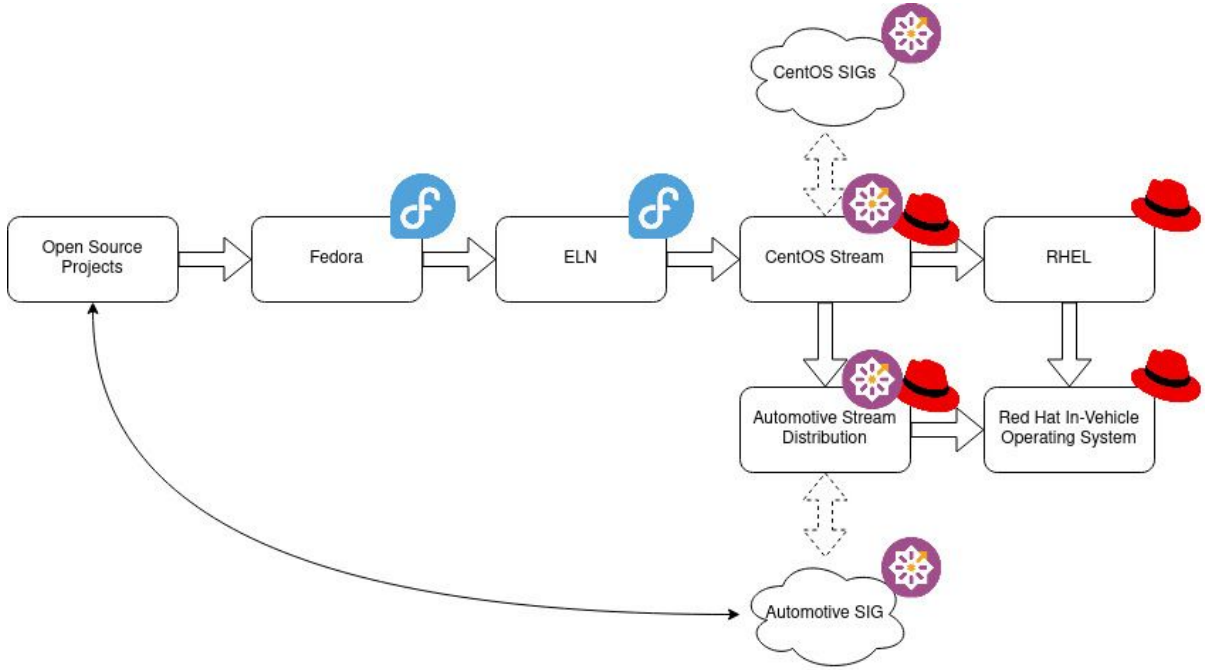
Automotive SIG

Automotive Stream Distribution (AutoSD)

- Upstream for Red Hat In-Vehicle OS (in dev)
- Skunkworks for projects like Eclipse BlueChi
- Compatible with emerging standards e.g. SOAFEE
- Nightly Arm, x86 builds
- Monthly calls, Matrix channel, mailing list

Upstream process

Upstream process similar to CentOS - RHEL:



Cloud SIG

Focuses on providing different cloud infrastructure applications that can be installed and run natively on CentOS Stream.

- RDO distribution of OpenStack
- OKD distribution of OpenShift/Kubernetes
- SCOS, CoreOS built from CentOS Stream

Docs SIG, Promo SIG, Artwork SIG

- Promo SIG promotes the CentOS project, handles outreach, manages social media, and plans events like CentOS Connect
- Artwork SIG creates artwork, like the new logo
- Docs SIG is newly chartered, a few things in the works:
  - Adapting upstream RHEL docs to CentOS
  - Updating/improving Contributors Guide
  - New simpler docs infra
  - Finding what old wiki content we need to migrate
  - Helping SIGs with their own docs

Q & A

🍪 Cookies 🍪

RPM PACKAGING WORKSHOP

CARL GEORGE
CPE EPEL Team Lead

@carlwgeorge@fosstodon.org
@carlwgeorge:matrix.org
carl@redhat.com

LAB: INITIALIZE

Open this link and click the “**Launch**” button.

bit.ly/hellorpm

WHAT IS RPM?

- Package format used by:
  - Fedora Linux
  - CentOS Stream
  - Red Hat Enterprise Linux
  - many others
- Consumed by package managers such as dnf

WHY PACKAGE WITH RPM?

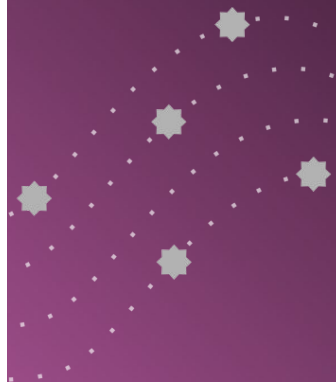
- Easily install, reinstall, remove, and upgrade software
- Query and verify installed packages
- Metadata to describe package properties and relationships with other packages
- Digitally signed packages to validate authenticity
- Distribute packages in dnf repositories
- Pristine sources to ease future maintenance

WHAT IS AN RPM PACKAGE?

- Special archive containing files and metadata
- Two main types
  - Binary RPMs contain files to be installed on the target system
  - Source RPMs contain software source code and instructions for building binary RPMs

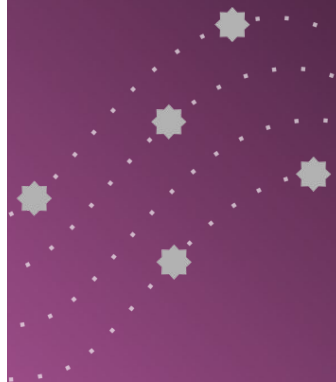
WHAT IS AN RPM SPEC FILE?

- Recipe for building the package
- Preamble that defines metadata about the package
- Body with several sections for various stages of the build process
- Conditionals for flexibility between operating systems, operating system versions, architectures, etc.



RPM MACROS

- Variables for text substitution in the spec file
  - Syntax: `%example` or `%{example}`
- Some macros accept parameters to influence the output
- Can be defined inside the spec file or on the system
  - `/usr/lib/rpm/macros.d/macros.*`
  - `/etc/rpm/macros.*`
  - `~/.rpmmacros`



RPM MACROS

- Can be conditional to only expand when the macro is defined
  - `%{?dist}`
- Another conditional form is to insert text when defined
  - `%{?rhel:--disable-feature}`
- Can be explored outside of the build process
  - `rpm --eval '%example'` → evaluate a specific macro
  - `rpm --showrc` → print all defined macros

COMMON MACROS

- Filesystem paths
  - `%{_bindir}` → `/usr/bin`
  - `%{_datadir}` → `/usr/share`
  - `%{_sysconfdir}` → `/etc`
- Operating system properties
  - `%{rhel}` → `9`
  - `%{dist}` → `.el9`
  - `%{el9}` → `1`

COMMON MACROS

- Build process helpers
  - `%autosetup` ⟶ extract source code archives and apply patches
  - `%configure` ⟶ `./configure` with packaging-specific options
  - `%make_build` ⟶ `make` with packaging-specific options
  - `%make_install` ⟶ `make install` with packaging-specific options

COMMON MACROS

- Legacy Python helpers
  - `%py3_build` → `python3 setup.py build`
  - `%py3_install` → `python3 setup.py install`
- Modern Python helpers
  - `%pyproject_wheel` → wheel-based Python build
  - `%pyproject_install` → wheel-based Python install

COMMON MACROS

- CMake helpers
  - `%cmake` → `cmake`
  - `%cmake_build` → `cmake --build`
  - `%cmake_install` → `cmake --install`
- Meson helpers
  - `%meson` → `meson`
  - `%meson_build` → `meson compile`
  - `%meson_install` → `meson install`

COMMON MACROS

- Test suite helpers
  - `%pytest` → `pytest`
  - `%ctest` → `ctest`
  - `%meson_test` → `meson test`

PACKAGING WORKSPACE SETUP

- `rpmdev-setuptree` (from the `rpmdevtools` package) creates several directories
  - `~/rpmbuild/BUILD`
  - `~/rpmbuild/RPMS`
  - `~/rpmbuild/SOURCES`
  - `~/rpmbuild/SPECS`
  - `~/rpmbuild/SRPMS`

LAB: PACKAGING WORKSPACE SETUP

Your first challenge is to set up your packaging workspace.

Click the “**Start**” button and follow the on screen instructions.

Once you have completed the instructions, click the “**Next**” button.

SPEC FILE PREAMBLE

- `Name` ⟶ name of the package, should match the spec file name
- `Version` ⟶ version of the software being packaged
- `Release` ⟶ used to distinguish between different builds of the same software version
- Together these properties form an identifier known as the NVR
  - `gawk-4.2.1-4.el8`
  - `tzdata-2024a-1.el9`
  - `virt-what-1.25-4.fc39`

SPEC FILE PREAMBLE

- `Epoch` → optional integer used to override normal version-release sorting order
  - Can never be removed
  - Last resort to correct upgrade path
  - `2024.01` > `1.0.0`
  - `2024.01` < `1:1.0.0`

SPEC FILE PREAMBLE

- `Summary` ⟶ short one line summary
- `License` ⟶ identifier for the license of the software
- `URL` ⟶ URL for more information about the software
- `BuildArch` ⟶ defaults to the build system architecture, can be set to `noarch` for packages with no architecture-specific files

SPEC FILE PREAMBLE

- `Source` → file name or URL of file needed to build the package, such as a source code archive or default configuration files
- `Patch` → file name or URL of patch to apply to the source code
- These two tags can be used multiple times
- Optionally suffixed with numbers
  - `Source0`
  - `Source1`

SPEC FILE PREAMBLE

- `BuildRequires` → other packages needed to build this package
- `Requires` → other packages needed to install this package
- `Recommends` → weak requires, installed by default but can be removed or skipped
- `Supplements` → reverse recommends

SPEC FILE PREAMBLE

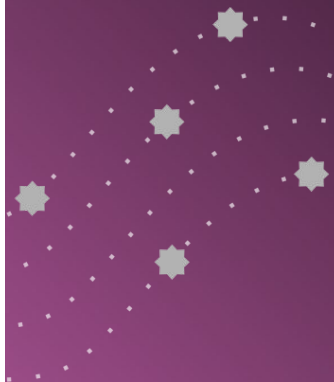
- `Conflicts` → other packages that cannot be installed at the same time
- `Obsoletes` → used to replace one package with another
- `Provides` → allows other packages to refer to this package by another name

SPEC FILE PREAMBLE

- `%description` → description of the package, can span multiple lines
- `%package <name>` → starts a preamble section for a separate package, often referred to as a sub-package
- `%description <name>` → description for a sub-package

SPEC FILE BODY

- `%prep` → commands to prepare the source code for building, such as unpacking archives and applying patches
- `%build` → commands to build the software
- `%install` → commands to copy the desired build artifacts into a directory tree relative to the `%{buildroot}`
- `%check` → commands to test the software, such as unit tests



SPEC FILE BODY

- `%files` → list of files and directories that will be installed on the target system
- `%changelog` → record of changes that have happened to the package between different versions and releases

FILE ATTRIBUTES

- In `%files`, each line can be preceded by an attribute
  - `%dir` → own just the directory itself, but not its contents
  - `%config` → mark as a configuration file
  - `%config(noreplace)` → mark as a configuration file and prevent it from being overwritten on updates
  - `%attr(<mode>,<user>,<group>)` → set non-default permissions or ownership

FILE ATTRIBUTES

- Some attributes accept relative paths, which copy the specified files into an appropriate path relative to the `%{buildroot}`
  - `%license` → copy files to `/usr/share/licenses/%{name}/` and mark as license files
  - `%doc` → copy files to `/usr/share/doc/%{name}/` and mark as documentation files

CREATING SPEC FILES

- From scratch
- Copy a similar spec file and adjust as needed
- Automatic templates from a text editor
- `rpmdev-newspec` (from the `rpmdevtools` package) will create a new spec file from templates

CREATING CHANGELOG ENTRIES

- By hand
- Copy another changelog entry and adjust as needed
- Text editor plugins
- `rpmdev-bumpspec` (from the `rpmdevtools` package) will create new changelog entries and simultaneously adjust the version and/or release tags

BUILDING RPMS

- RPMs are built with the `rpmbuild` command
  - `rpmbuild` expects the directory structure from `rpmdev-setuptree`
- Various build modes
  - `-bs` → build an SRPM from a spec file and sources
  - `-bb` → build an RPM from a spec file and sources
  - `-ba` → build both an SRPM and an RPM from a spec file and sources
  - `--rebuild` → build an RPM from an SRPM

QUALITY CHECKING RPMS

- `rpmlint` is a linter tool for spec files, SRPMs, and RPMs
- Identifies common packaging errors
- Ideal to resolve all errors and warnings, but not always possible

QUALITY CHECKING RPMS

- `rpm` can query an uninstalled RPM by using the `--package` flag
- Consider the following additional flags:
  - `--info`
  - `--list`
  - `--requires`
  - `--provides`
  - `--conflicts`
  - `--changelog`

LAB: PACKAGING BELLO

Your next challenge is to package `bello`, a program written in Bash.

Click the "**Start**" button and follow the on screen instructions.

Once you have completed the instructions, click the "**Next**" button.

INSTALLING BUILD REQUIREMENTS

- `rpmbuild` needs the build requirements listed in the spec file to be installed on the build host
- Can be installed manually or with `dnf builddep`

LAB: PACKAGING CELLO

Your next challenge is to package `cello`, a program written in C.

Click the “**Start**” button and follow the on screen instructions.

Once you have completed the instructions, click the “**Next**” button.

LAB: PACKAGING PELLO

Your next challenge is to package `pello`, a program written in Python.

Click the "**Start**" button and follow the on screen instructions.

Once you have completed the instructions, click the "**Next**" button.

MOCK

- Drawbacks of using `rpmbuild` directly
  - Build requirements installed directly on build host
  - Build requirements that happen to already be installed are easy to forget to include in the spec file
  - Can only build RPMs targeting the same operating system and operating system version as the build host

MOCK

- `mock` is a tool that builds RPMs in isolated chroots
  - Uses `rpmbuild` internally
  - Build requirements are installed in the chroot, not the build host
  - Helps identify missing build requirements
  - Can build RPMs targeting a different operating system and operating system version as the build host
  - Chroots are automatically created and removed
- Widely used (`koji`, `copr`, `fedpkg`, and more)

LAB: BUILDING WITH MOCK

Your final challenge is to build the `pello` package again, but using the `mock` tool this time.

Click the "**Start**" button and follow the on screen instructions.

Once you have completed the instructions, click the "**Next**" button.

BECOME A FEDORA/EPEL PACKAGER

Interested in doing more? Consider becoming a Fedora and EPEL package maintainer.

bit.ly/fedorapackager

THAT’S ALL FOLKS

@carlwgeorge@fosstodon.org
@carlwgeorge:matrix.org
carl@redhat.com