Building Fedora CoreOS at Nest with Fedora 2022

Renata Ravanelli - Red Hat
Senior Software Engineer
 renata.ravanelli@gmail.com
 # ravanelli on libera.chat

Saqib Ali - Red Hat
Software Engineer Intern
 saqibsali00@gmail.com
 # saqali on libera.chat
Today's Talk

- History
- What is Fedora CoreOS?
- Red Hat CoreOS Vs Fedora CoreOS
- Multiple Update Streams
- Why learn how FCOS is built?
- Build Process
- Components of the “config”
- CoreOS-Assembler (cosa)
- Overrides and New Packages

- What about testing?
- How do we deliver FCOS?
- Demos
- Challenge!
- Get involved!
- Your Questions!
History

Fedora, Container Linux and Atomic host

Container Linux
- Philosophy: automatic updates
- Provisioning Stack,
- Immutable infra
- Cloud Native Expertise

Atomic Host
- Fedora Foundation, base OS
- and its structure such as packages and Kernel
- Update Stack
- Selinux Enhanced Security

Came from

Fedora, Container Linux and Atomic host

=  Container Linux + Atomic Host
What is Fedora CoreOS?
Fedora CoreOS is an automatically-updating, minimal operating system for running containerized workloads securely and at scale. It is currently available on multiple platforms, with more coming soon.
Red Hat CoreOS Vs Fedora CoreOS

Based on RHEL;
Is only meant to be used with OpenShift;
Red Hat CoreOS is not a standalone OS, it is a component of OpenShift;
Automated provisioning via ignition;
SELinux enforcing;
Updates and configuration controlled by cluster operator;
RPM-OStree technology

Based on Fedora;
Standalone OS with automatic updates (Reliable updates);
Automated provisioning via ignition;
SELinux enforcing;
Podman or moby engine container runtimes by default;
Can work as part of a cluster with OKD;
Share components and tooling with RHEL CoreOS;
RPM-OStree technology
Multiple Update Streams

Fedora CoreOS is available across 3 different release streams:

**Stable**
- **v 35.20220424.3.0**
- **JSON — 2 days ago**

The Stable stream is the most reliable version of Fedora CoreOS. Releases are battle-tested within the Testing stream before being promoted.

**Testing**
- **v 36.20220505.2.0**
- **JSON — 2 days ago**

The Testing stream contains the next Stable release. Mix a few Testing machines into your cluster to catch any bugs specific to your hardware or configuration.

**Next**
- **v 36.20220507.1.0**
- **JSON — 2 days ago**

The Next stream represents the future. It provides early access to new features and to the next major version of Fedora. Run a few Next machines in your cluster, or in staging, to help find problems.
<table>
<thead>
<tr>
<th>Platform</th>
<th>Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alibaba Cloud</td>
<td>ppc64le, x86_64</td>
</tr>
<tr>
<td>AWS</td>
<td>s390x, x86_64</td>
</tr>
<tr>
<td>DigitalOcean</td>
<td>x86_64</td>
</tr>
<tr>
<td>GCP</td>
<td>x86_64</td>
</tr>
<tr>
<td>Nutanix</td>
<td>x86_64</td>
</tr>
<tr>
<td>OpenStack</td>
<td>x86_64</td>
</tr>
</tbody>
</table>

**AWS**
- **stable**
- **Regions**
  - **us-east-1**
- **Release**: 36.20220716.3.1
- **Image**: ami-03929f88dfb4b1c1c

**GCP**
- **stable**
  - **Project**: fedora-coreos-cloud
  - **Family**: fedora-coreos-stable (details)
Why learn how FCOS is built?

- Build FCOS yourself!
- Build a custom FCOS-like OS
- Learn about the components that make up FCOS
Build Process

FCOS Config
- rpm-ostree treefil
- Repos
- Misc other bits

coreos-assembler container
- Fedora Base
- Build Script Installed

coreos-assembler repo
- Build Script
- Dockerfile

Volume mounted in
- Disk Images
- ostree commit
CoreOS-Assembler (cosa)

tooling to build the OS

• containerized collection of tools used to build FCOS-style systems
• serves both local development and production level build systems
• Built images found at quay.io/coreos-assembler/coreos-assembler
Components of the “config”

**manifest.yaml**
tells RPM-OSTree how to generate OSTree commits with list of RPMS

**overlay.d/**
additional information that is layered in the OSTree commit

**image.yaml**
configuration of the final disk images
For generating OSTree commits, cosa uses manifest.yaml

It is a list of RPMs and a set of rpm-md repositories they come from. It also supports postprocess to make arbitrary changes
overlay.d/

Components of the “config”

- subdirectories are added to OSTree commit
- used to modify default configuration (ie. disable SSH passwords)

```
ravanelli-redhat:overlay.d ravanelli$ ls
05core 09misc 15fcos 16disable-zincati-and-pinger 35coreos-iptables
08nouveau 14NetworkManager-plugins 16disable-zincati 20platform-chrony
ravanelli-redhat:overlay.d ravanelli$ cat 15fcos/etc/ssh/sshd_config.d/40-disable-passwords.conf
# Disable password logins by default.
# https://github.com/coreos/fedora-coreos-tracker/issues/138
# This file must sort before 50-redhat.conf, which enables
# PasswordAuthentication.
PasswordAuthentication no
```
image.yaml

Components of the “config”

- Supports customization of disk images
- Provides default “opinionated” settings

```yaml
This file contains defaults for image.yaml

bootfs: "ext4"
rootfs: "xfs"
# Add arguments here that will be passed to e.g. mkfs.xfs
rootfs-args: ""

# Additional default kernel arguments injected into disk images
extra-kargs: []

# Can also be oci-chunked
ostree-format: oci
# True if we should use `ostree container image deploy`
deploy-via-container: false

# Set this to a target container reference, e.g. ostree-unverified-reg
# container-imgref: ""

# Format used when generating a squashfs image. Can also be e.g. gzip
squashfs-compression: zstd

# Defaults for VMware OVA, matching historical behavior
vmware-hw-version: 13
vmware-os-type: rhel7_64Guest
```
Overrides and New Packages

● Development Overrides:
  ○ In cosa you can do it via overrides. There are two subdirectories of overrides: **overrides/rootfs** and **overrides/rpm**
  ○ When you are hacking/testing a build that is the easier way to change and test packages and other configurations, especially because you won't need to care with repositories for example.
  ○ In fedora-coreos-config there is the **manifest-lock.overrides.yaml**. You can also override a package using the manifest file.

● New packages:
  Should be added in **manifest-lock.x86_64.json** for the specific architecture and **fedora-coreos-base.yaml**
What about testing?

Compiled in Kola More complex tests written in Golang

External Tests Bash scripts that live alongside the config

```bash
#!/bin/bash
# kola: { "exclusive": false }
# We can run this on both FCOS and RHCOS as neither should have a zram device
# enabled by default. (In RHCOS, there is no zram support at all)
set -xeuo pipefail

. $KOLA_EXT_DATA/commonlib.sh

# make sure we don't default to having swap on zram
# https://github.com/coreos/fedora-coreos-tester/issues/509
# https://github.com/coreos/fedora-coreos-config/pull/687
if [ -e /dev/zram0 ]; then
    fatal "zram0 swap device set up on default install"
fi
ok no zram swap by default
```
cosa kola
the testing framework

- Local testing with QEMU
- Supports testing with multiple cloud providers (AWS, Azure, GCP, OpenStack)
- Nifty features
  - ignition configs for each test
  - reboots
  - reruns and timeouts
  - put multiple tests in one VM
- As simple as “cosa kola run”
How do we deliver FCOS?
How are we able to ship 3 streams and 3 architectures every 2 weeks?

Investment in CI/CD
None of this happens without... Passing Tests!

<table>
<thead>
<tr>
<th>Status</th>
<th>Platform</th>
<th>Time</th>
<th>Time</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>kola-aws</td>
<td>13 hr - #325</td>
<td>4 days 0 hr - #312</td>
<td>1 hr 11 min</td>
</tr>
<tr>
<td>☑</td>
<td>kola-azure</td>
<td>13 hr - #80</td>
<td>12 days - #57</td>
<td>45 min</td>
</tr>
<tr>
<td>☑</td>
<td>kola-gcp</td>
<td>13 hr - #169</td>
<td>3 days 14 hr - #164</td>
<td>15 min</td>
</tr>
<tr>
<td>☐</td>
<td>kola-kubernetes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>☑</td>
<td>kola-openstack</td>
<td>13 hr - #340</td>
<td>1 day 10 hr - #337</td>
<td>45 min</td>
</tr>
</tbody>
</table>
Versatile Tooling: cosa

- cosa container has all the tooling to build and test
  - building
  - testing
  - compressing builds
  - uploading build
  - editing build metadata
  - and so on....
- easy to launch tests on major cloud providers
- same cosa for local tests or production builds
Lockfiles
controlling package versions

- “lockfiles” allow us to control the package versions for each stream
- jenkins job “bumps” lockfile package versions after test run
Overrides: Lockfiles

controlling package versions

- Lockfiles are flexible with overrides
- Override the latest available package version
  - Pinning: use an older version
  - Fast-track: use a new package not yet available through fedora stable channels
Multi-arch Builders

building multiple architectures

- After the build job for x86_64 passes it triggers the multi-arch builds
- One build job is created for each architecture
- Multi-arch builds are farmed out to individual nodes of that architecture running Fedora CoreOS
- We use podman remote to access other architectures. It means only one Jenkins instance is responsible for all builds
- Same process is done for all

In the future: ppc64le
Demos

Override Kernel packages via cosa and manifest.yaml
Demos

Adding a new package
Challenge! Try out Fedora CoreOS and join our Community!
Get involved!

- Web: https://getfedora.org/coreos
- Issues: https://github.com/coreos/fedora-coreos-tracker/issues
- Forum: https://discussion.fedoraproject.org/tag/coreos
- Docs: https://docs.fedoraproject.org/en-US/fedora-coreos/
- Mailing list: coreos@lists.fedoraproject.org
- IRC: libera.chat #fedora-coreos
- Matrix #coreos:fedoraproject.org

Go checkout the tutorials:
Questions
Thank you!