datto

Adapting to AppStreams

Delivering the Datto Linux Agent for RHEL 8

Neal Gompa

Who am I?

- Professional technologist
- Linux user for nearly fifteen years
- Contributor and developer in Fedora, Mageia, openSUSE, and OpenMandriva Linux distributions
- Contributor to RPM, DNF, and various related projects
- DevOps Engineer at Datto, Inc.

datto

All About Datto

Founded in 2007

23 offices around the world

1,800+ employees worldwide & growing

17,000+ managed service provider partners

100% channel only

What We Offer

Datto products empower our community of Managed Service Provider partners with the right technology, business tools, and support to enable each and every one of their customers to succeed. It's an approach that has made us the world's leading innovator of MSP-delivered IT solutions.

Growth Products

Unified Continuity

Reliable data protection for full IT environment maximizing up time

- SIRIS
- ALTO
- Datto Cloud Continuity for PCs
- Datto File Protection
- NAS
- SaaS Application Protection
  - Office 365
  - Google Suite

Networking

Fully cloud managed networking solutions designed for MSPs

- Datto Networking WiFi
- Datto Networking Switches
- Datto Networking Edge Routers
- Datto Managed Power

File Sync & Share

Fully managed File Sync & Share solution

- Datto Workplace

Efficiency Products

Professional Services Automation (PSA)

SaaS platform for MSPs to manage their entire business

Remote Monitoring & Management (RMM)

Cloud-based Software for MSPs to manage SMB endpoints

datto

Datto Linux Agent?

Datto Linux (Backup) Agent

- Part of our Business Continuity/Disaster recovery (BCDR) solution
- Enables seamless backups of x86_64 Linux systems
- Components:
  - `dattobd`: open source kernel module
  - `dlad`: proprietary userspace daemon

Datto Linux (Backup) Agent

- More than 300 releases of the kernel and user-space components in the past five years
  - A little under a third of those have been releases to customers
- Over 50 Linux distribution releases have been supported across all versions of DLA for a range of Linux distributions
  - We support slightly under half that with the latest DLA versions

How we build the Datto Linux Agent

datto

Open Build Service

The Open Build Service (OBS) is a software solution created by SUSE to build and manage the openSUSE and SUSE Linux Enterprise distributions. It's similar to Koji, the RHEL/Fedora build system.

However, it was designed from the beginning to support a wide variety of Linux based platforms. Notably, it can build packages, repositories, and images for Red Hat/Fedora, SUSE, and Debian/Ubuntu systems.

SUSE offers a hosted version as the openSUSE Build Service, and the appliance image is freely available for you to set up your own.

Why we use the Open Build Service?

- Source input flexibility through “source services” that allow scripted retrieval and processing of sources
- Easy scaling of resources through OBS workers that detect the master and auto-connect
- Automatic reverse dependency rebuilding on package updates to ensure dependencies are linked correctly
- Easy to deploy and get started with using the official appliance provided on the website
- Lets us build packages natively for RPM and Debian distributions using RPM spec files (using debbuild for Debian/Ubuntu)

And everything Just Worked…?

OBS and AppStreams

- Application Streams was introduced to Red Hat Enterprise Linux with the RHEL 8 beta in November 2018
- Unfortunately, the incomplete state of the public beta made it difficult to do any development
- Everything was pushed back to after RHEL 8 release
- No guidance on how to build AppStreams either…
- Thus, OBS had no support for this when RHEL 8 released

Go to the source!

Applications Streams is the implementation

of the upstream Fedora Modularity project in

Red Hat Enterprise Linux

Fedora Modularity guidance…?

Module builds in Fedora, both locally and remotely, depend on a service called the Module Build Service, which processes the modulemd YAML files to generate a build environment.

In order to do this, it needs to talk to a Koji server to figure out what is available and create the set of packages that should exist in the build environment (In Koji terms: a build root).

This implies that we should be running a Koji instance to build packages depending on modules. It also implies that we have access to the Koji instance building the target distribution.

We have OBS, not Koji...

So now what?

Three potential options

- Mirror the distribution and de-modularize the repositories
- Deploy Koji and MBS and modify it to work purely off repositories
- Enhance OBS to handle modular content

Mirror distribution and de-modularize

Pros

- Makes building packages with any tooling possible
- Reduces the amount of effort to adapt tooling to support AppStreams in distributions

Cons

- Huge outlay of storage required (hundreds of gigabytes over time)
- Process has to be repeated every time content is mirrored, making it slow and expensive to update distribution content
- Loss of modular dependency semantics, making it possible to accidentally create invalid dependency chains at runtime

Deploy Koji and MBS

Pros

- Leverages the same build pipeline technologies that Red Hat and Fedora use
- Well-understood technology path for building packages
- Easy to build our own AppStreams with

Cons

- Forces maintenance of duplicate infrastructure
- Would require synchronizing between Koji and OBS to have a complete store of package content
- Unequal capabilities between the two systems may cause more trouble than it is worth

Enhance OBS to support AppStreams

Pros

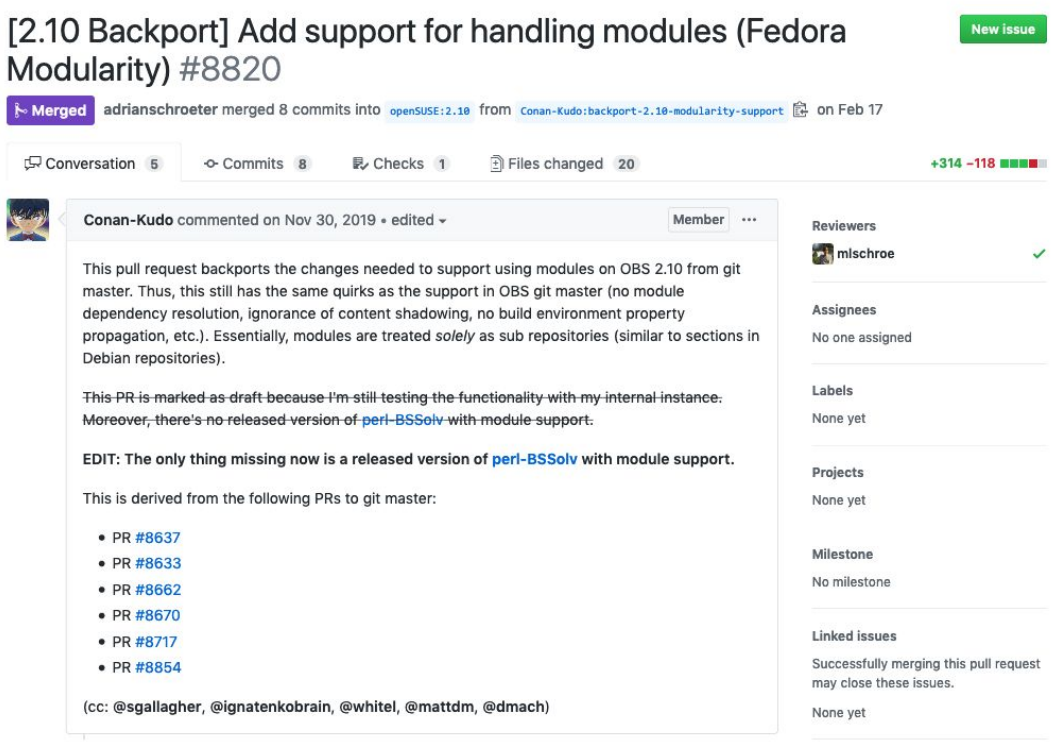
- Leverages our existing tooling and pipelines
- Does not require supporting duplicate infrastructure

Cons

- Producing AppStreams sanely would not be straightforward to implement
- Build system resolver needs to be taught concepts regarding AppStreams

What did we do?

- Initially pursued de-modularization strategy using GrobiSplitter from Fedora Modularity, as running a second build system for just one Linux distribution was not appealing
- I met with the OBS team at the openSUSE Conference along with members of the DNF/YUM team to hash out a strategy to support AppStreams in OBS
- The upstream OBS project implemented some of this over the course of last year, which led us to refocus on porting that to the stable OBS release



datto

We’re done now, right?

After upgrading to OBS 2.10.1 (which included the backported feature), we just needed to add a snippet to our project configuration to enable the `virt` module with the `rhel` stream. This is shown on the right.

This enabled the AppStream in our build environment and allowed everything to work, mostly...

Snippet from OBS project configuration:

```
%if "%{_repository}" == "CentOS_8"
# for libiscsi-devel for libagent
ExpandFlags: module:virt-rhel
%endif
```

datto

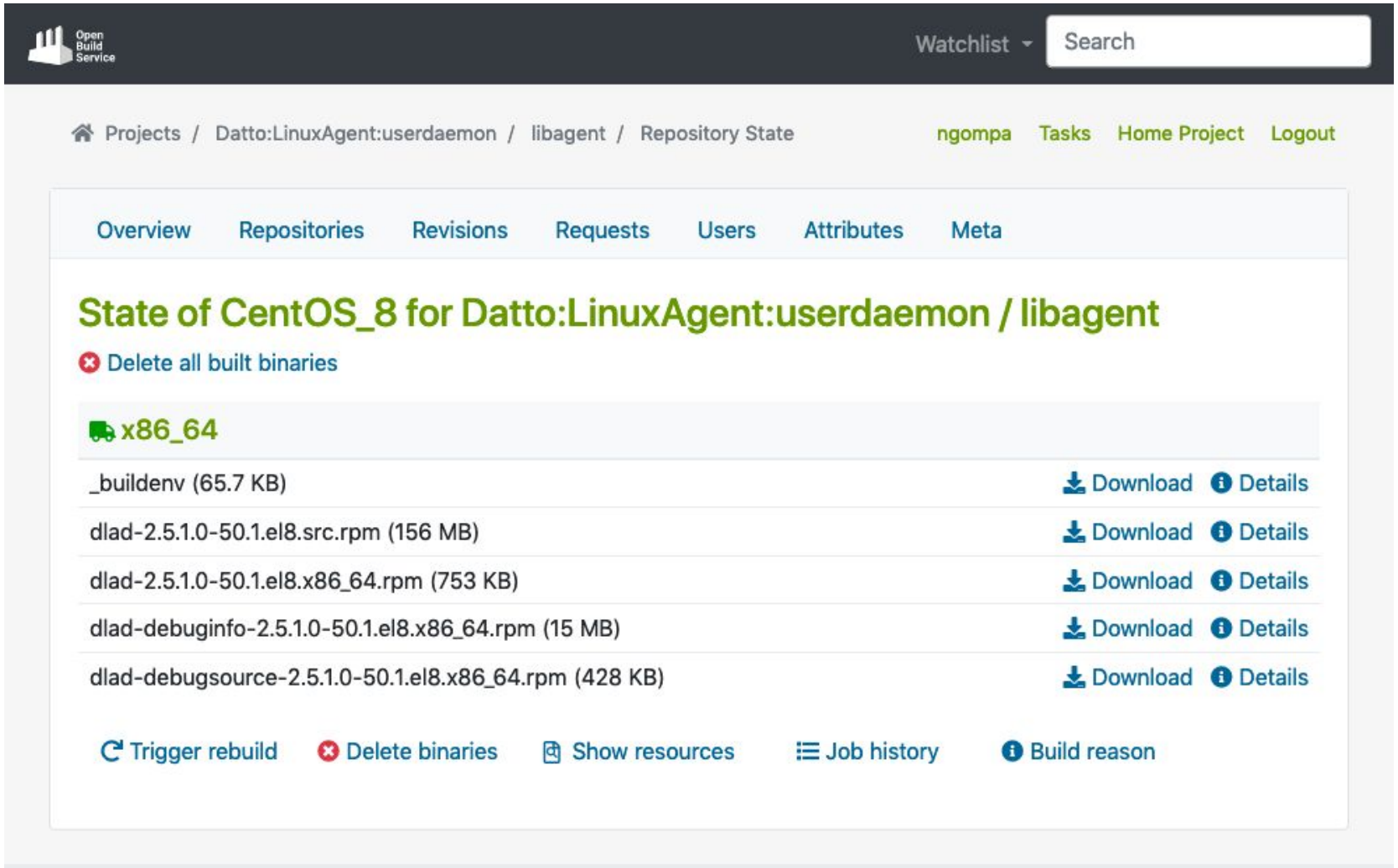
One last thing...

As it turned out, there was one remaining problem: a missing development header package. Specifically, `dlad` uses `libudisks2`, and we needed `libudisks2-devel` to link to it.

This was easily rectified: we filed a case with Red Hat in January and had it added to the CodeReady Builder repository for RHEL 8.3. But in order to get things out the door now, we needed it faster...

CentOS Stream was the answer here, and we had it added there so that our build system could consume it now and be ready faster.

And now we've done it!



datto

Special Thanks

- Stephen Gallagher from Fedora Modularity Team
- Daniel Mach and Jaroslav Mracek from DNF Team
- Adrian Schröter and Michael Schröder from Open Build Service Team

Resources

- GrobiSplitter: https://github.com/fedora-modularity/GrobiSplitter
- Module Building in a Box (Koji + MBS): https://github.com/fedora-infra/mbbox
- Fedora Modularity: https://docs.pagure.org/modularity/
- Koji: http://koji.build/
- Open Build Service: https://openbuildservice.org/

Questions?

ngompa@datto.com

datto

The world’s leading provider of
MSP-delivered IT solutions