Packaging GitLab using omnibus-ruby

Taking installation instructions from 400 lines to 3 lines

Jacob Vosmaer

GitLab B.V.

DevOps Amsterdam meetup, 30 September 2014

Introduction

The omnibus-gitlab story

- Challenge: GitLab was hard to install and maintain
- Solution: imitate Chef Server
- Unexpected benefits

Technical topics

- Dependencies: to bundle or not to bundle
- Omnibus-gitlab architecture
- Consistent service management with Runit
- Designing for DRBD
- Omnibus-gitlab and security

GitLab and GitLab B.V.

- **GitLab** is open source software to collaborate on code.
- MIT open source license, 10k+ commits and 500+ contributors, lead author is Dmitriy Zaporozhets (UKR)
- Used by individuals, small companies, government agencies and billion-dollar corporations.
- **GitLab B.V.** was co-founded by Sytse Sijbrandij (NL) and Dmitriy
- GitLab B.V. offers GitLab Enterprise Edition, support subscriptions and consulting services.
- We also offer a free GitLab SaaS at `https://gitlab.com`
- My name is Jacob Vosmaer and I am a Senior Service Engineer at GitLab B.V.
- The views I present here are my own and may or may not reflect those of my employer.

Getting to know you

Who has ever . . .

- installed software on Linux?
- installed software from source?
- installed a Ruby on Rails (or similar) application from source?
- shipped software for others (users) to install?
- created their own RPM or Debian packages?

Challenge: GitLab was hard to install and maintain

Installing GitLab in January 2014.

- Install dozens of packages (Debian build-essential etc.)
- Compile Ruby and Git from source
- Orchestrate the integration between five daemons
- Install an init script
- Official documentation was only for Ubuntu 12.04

To make things worse, falling behind on the monthly GitLab release schedule could quickly get painful for administrators.

Looking for solutions

- 'Native' Ubuntu packages seemed problematic for dependencies (we were already compiling Ruby ourselves) and hard
- Docker: too immature; installing GitLab should not require a back-ported kernel and preferably no EPEL either
- Omnibus-ruby: proven for an app of similar complexity (Chef Server) and uses familiar tools (Ruby, Chef)

What is omnibus-ruby

- `https://github.com/opscode/omnibus`
- Packaging tool created by Chef, Inc. to distribute the Chef Client and Chef Server applications
- Monolithic platform-specific packages with all dependencies compiled from source at build time
- Chef Server uses Runit and Chef Solo to manage and configure the server application after it is installed
- Open source, Apache license

From omnibus-chef-server to omnibus-gitlab: a success!

Based on the technology overlap between Chef Server and GitLab and our familiarity with Ruby and Chef we decided to adapt **omnibus-chef-server**, the 'build lab' for Chef Server, into **omnibus-gitlab**.

- First pre-release of omnibus-gitlab on 14 February 2014 after 257 commits of copy & paste, search & replace and head scratching.
- Users could now install and configure GitLab in only three steps: download, install and reconfigure.
- We went from 400 lines of installation instructions to 3 lines.

Unexpected benefits

Although we knew nothing about packaging, we got to copy the design of people who did. Thank you Chef!

- Supporting new distributions became MUCH easier: Centos 6 support took `3 files changed, 16 insertions(+), 5 deletions(-)`
- Improved security through more restrictive file and directory permissions
- A directory structure organized for DRBD-based High Availability
- We now run our gitlab.com SaaS with omnibus packages

Disadvantages

- Compared to a GitLab installation from source, users have less control. We are still adding configuration options where needed based on the demands of our users and ourselves (gitlab.com SaaS).
- GitLab, as a software project, gained $\sim$ 5000 lines of code.
- The monthly GitLab releases take more effort now.

Introduction

The omnibus-gitlab story

- Challenge: GitLab was hard to install and maintain
- Solution: imitate Chef Server
- Unexpected benefits

Technical topics

- Dependencies: to bundle or not to bundle
- Omnibus-gitlab architecture
- Consistent service management with Runit
- Designing for DRBD
- Omnibus-gitlab and security

Dependencies: to bundle or not to bundle

- Traditional Debian/RedHat packages use dependent packages to avoid installing software twice
- Omnibus 'monolithic' packages avoid dependent packages and bundle their dependencies
- Traditional packages are space-efficient and allow for 'global' security updates: fix OpenSSL once for the entire server
- Disadvantages include lowest common denominator versions and inconsistent dependency versions across Linux distributions and OS versions
- Monolithic packages make life easier for developers
- When done right, there is a lower risk of breaking monoliths with `apt-get upgrade`
- Omnibus-gitlab installs 900MB of files
- A security release for a bundled dependency means a security release for the monolith

Omnibus-gitlab architecture

- **Building**: Use omnibus-ruby to download, compile and install GitLab and all its dependencies on a build machine matching the deployment platform
- **Configuring**: Use chef-solo to create users, write dependency configuration files, create data directories etc.
- **Managing**: Use Runit to start, stop and restart daemon processes.
- Configuration and management is done via **omnibus-ctl**, a combined front-end for chef-solo and Runit.

Consistent service management with Runit

- **Runit** is 'a UNIX init scheme with service supervision'
- 'supervision' means automatic restarts of services
- Services are arranged in a supervision tree using parent-child process relations
- Simple, powerful and reliable
- The documentation is good but terse (there are only a few pages but you have to read them ten times)
- Omnibus-gitlab installs its Runit grandparent process via Upstart/Systemd/SysV init; Runit handles the actual omnibus-gitlab daemons.
- This is a big win: we get consistent service management across all Linux distributions we support with omnibus-gitlab
- Another piece of good design that Omnibus-gitlab inherited from Chef Server

Designing for DRBD

- After working on omnibus-gitlab I was assigned to adapt our existing Chef cookbook to create a DRBD-based Highly Available GitLab deployment
- Lots of cursing ensued as I tried to get all stateful services to write their data on the DRBD-replicated filesystem
- Then I realized this problem had already been solved by Omnibus-gitlab: all application state was being written in directories under `/var/opt/gitlab`, so all we had to do was mount a filesystem there and replicate it
- Another piece of good design that Omnibus-gitlab inherited from Chef Server

Omnibus-gitlab and security

- Thanks to Chef Inc. promptly updating their OpenSSL build script we had **new Omnibus-gitlab packages** within 24 hours of the announcement of Heartbleed (OpenSSL CVE-2014-0160).
- In the past 8 months we had one issue where the design we copied from Chef Server was insecure. When we became aware of the problem we contacted Chef and did a **joint** security **release** to address this.
- One security issue so far in omnibus-ruby, the build tool that creates the actual .deb and .rpm packages. Chef fixed the issue within days of our report with a **security release**.
- All in all, imitating Chef Server has raised the bar for security of GitLab deployments

Thank you for listening!