Building and Distributing The Kolide Launcher

QueryCon 2019

Joseph Sokol-Margolis

SRE

seph @ kolide.co

github.com / directionless

seph @ osquery slack

twitter.com / twseph

KOLIDE

About Me

SRE and Infrastructure at Kolide

Osquery community member.

Previously at Fastly, Twitter, and ActBlue

Enjoys data, infrastructure as code, and simplification

Why are we doing this anyhow?

Kolide Is

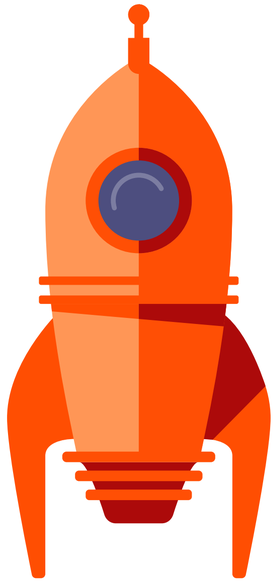
- A Company
- Open Source Tools: Fleet & Launcher
- SaaS Offering: ~~Cloud~~, K2
- Making it ***simple*** to deploy user focused security tools

What is simple?

- Don’t surprise people — Follow the underlying platform norms
- Use native packages. No `curl | bash`
- Avoid security prompts. Sign binaries and packages
- Handle updates. Ensure current versions
- Enroll hosts on installation. Minimize user steps

Launcher is

- How we achieve *simple*
- Kolide's osquery endpoint software
- Supplement osquery's data
- Integrates with platform service management
- Manages updating osquery and itself



Endpoint Enrollment

To make company wide deployment simple, we ship customer-specific packages.

These contain an *enroll* secret.

This secret is coordinated with the server.

Endpoint Enrollment

To make company wide deployment simple, we ship customer-specific packages.

These contain an *enroll* secret.

This secret is coordinated with the server.

This requires a lot of packages

So Many Packages

- Specific to each signup (We've had about 2,000)
- Specific to each platform (Mac, Windows, RedHat, Debian)
- Rebuilt for each launcher version
- Rebuilt for each osquery version
- That's 8,000 packages for each version update

Goals

KOLIDE

Goals

- Build a lot of packages
- Minimize staff toil and cognitive overhead
- Faster release cycle and fewer bugs
- Less fear around releases
- Create happy customers

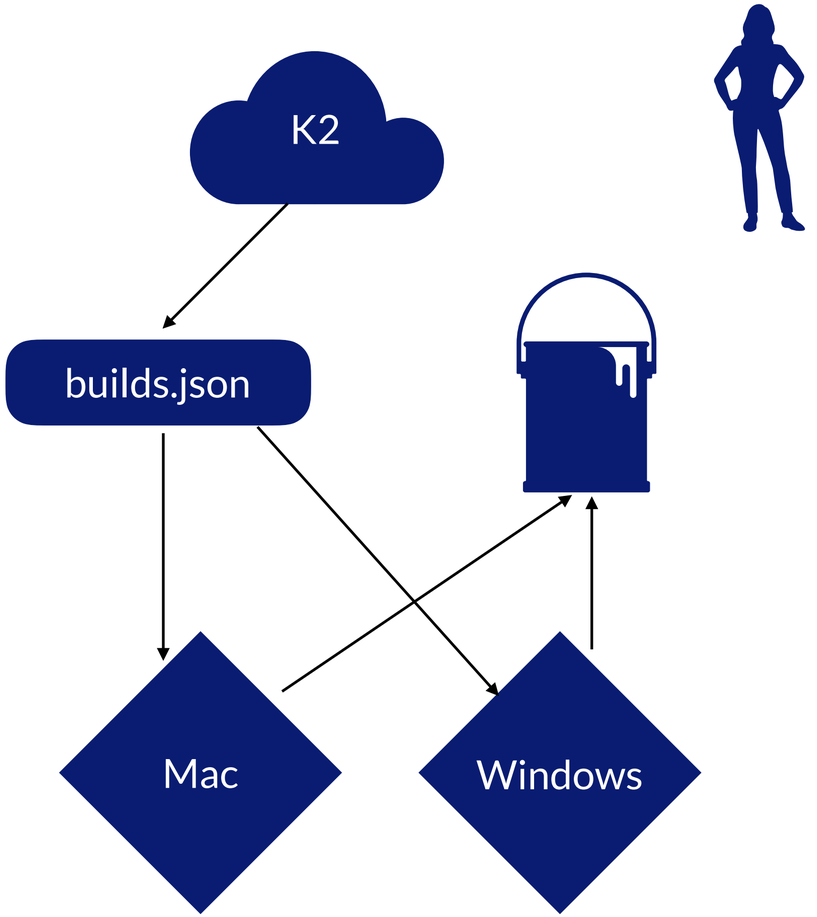
How do we achieve that?

Tooling Overview

KOLIDE

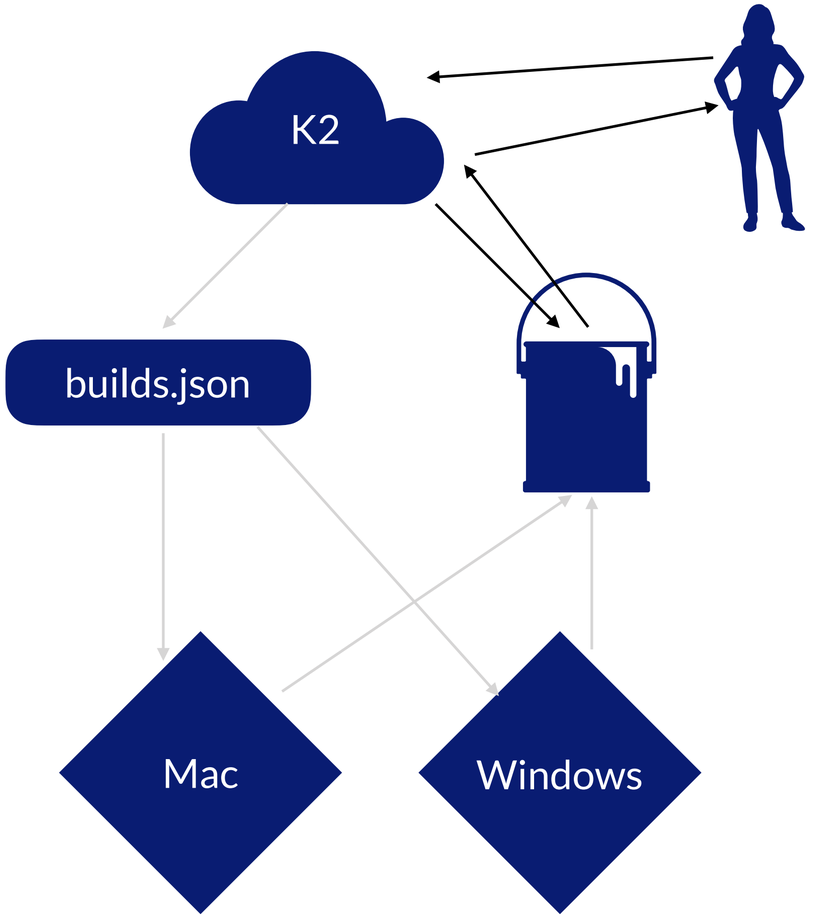
Automation Flow

1. K2 cron job creates builds.json
2. Workers:
    1. read builds.json
    2. Build packages
    3. Upload to bucket



Download Flow

1. User requests package
2. K2 checks bucket
3. K2 returns signed bucket URL



Underlying Tools

- Launcher's `pkg/packagekit`
- Launcher's `pkg/packaging`
- Launcher's `cmd/package-builder`
- kworker

Gory Package Details

- Packages are trying to put files onto disk. Most packaging tools convert a directory to the platform format.
- Scripts (post-install, pre-remove, etc) are often packaged as metadata.
- Init systems each need their own config files and scripts
- Windows is different. No files, per se, it’s all a database of objects.
- **packagekit provides a unified set of tools to work with these**

package-builder

- Building launcher & osquery packages for ***platform-init-package*** triples
- Via packagekit, uses os-native tooling, for simple code signing
- Builds linux packages through docker
- Flags for what launcher configuration options it creates

Lots of Flags

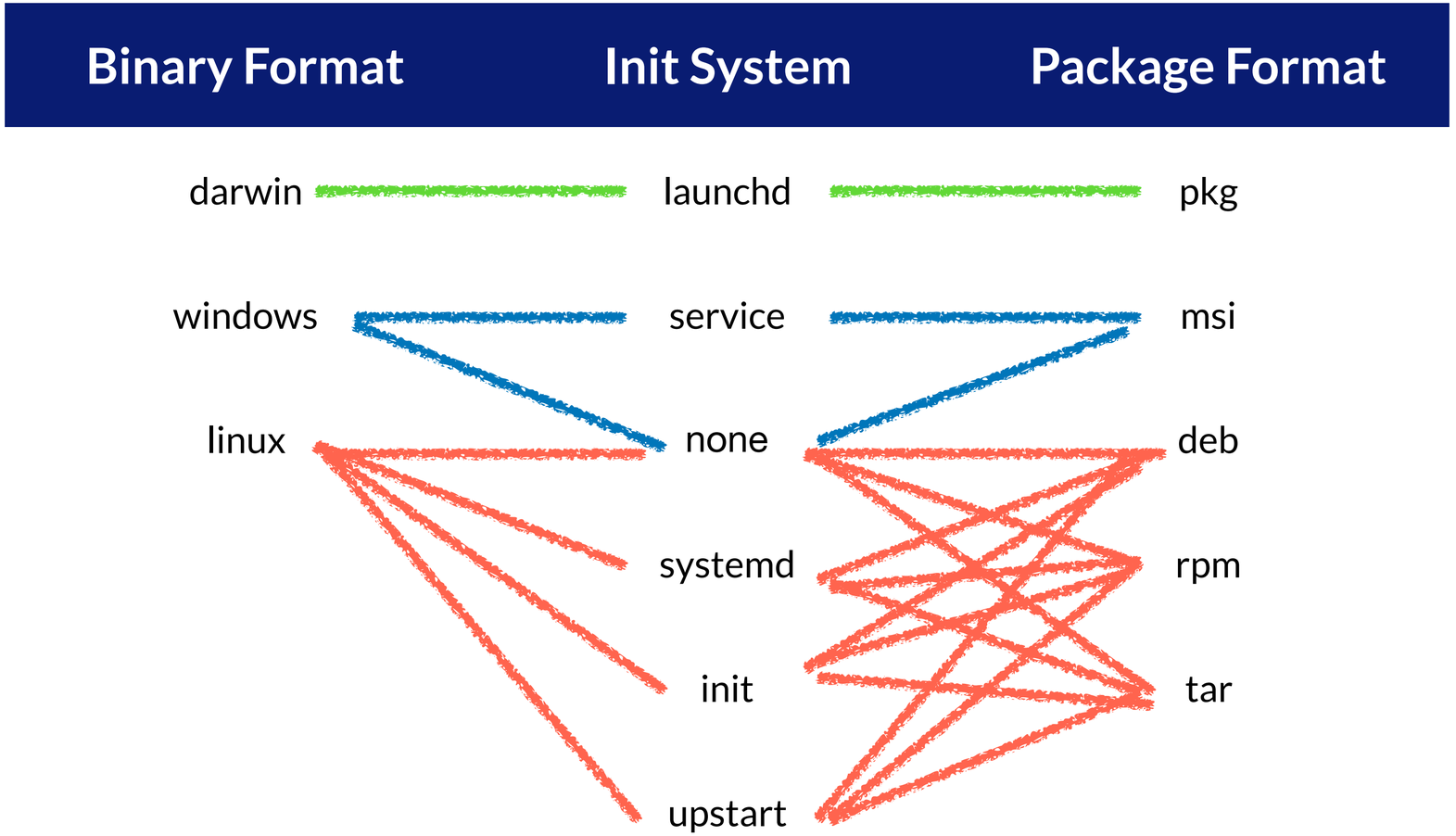
```
USAGE
  package-builder make [flags]

FLAGS
  -autoupdate false              whether or not the launcher packages should invoke the launcher's --autoupdate flag
  -cache_dir                     Directory to cache downloads in (default: random)
  -cert_pins                     Comma separated, hex encoded SHA256 hashes of pinned subject public key info
  -control_hostname              the value that should be used when invoking the launcher's --control_hostname flag
  -debug false                   enable debug logging
  -disable_control_tls false     whether or not the launcher packages should invoke the launcher's --disable_control_tls flag
  -enroll_secret                 the string to be used as the server enrollment secret
  -extension_version stable      What TUF channel to download the osquery extension from. Supports filesystem paths
  -hostname                      the hostname of the gRPC server
  -identifier launcher           the name of the directory that the launcher installation will shard into
  -insecure false                whether or not the launcher packages should invoke the launcher's --insecure flag
  -insecure_transport false      whether or not the launcher packages should invoke the launcher's --insecure_transport flag
  -launcher_version stable       What TUF channel to download launcher from. Supports filesystem paths
  -mac_package_signing_key       The name of the key that should be used to packages. Behavior is platform and packaging specific
  -omit_secret false             omit the enroll secret in the resultant package (default: false)
  -osquery_version stable        What TUF channel to download osquery from. Supports filesystem paths
  -output_dir                    Directory to output package files to (default: random)
  -package_version               the resultant package version. If left blank, auto detection will be attempted
  -root_pem                      Path to PEM file including root certificates to verify against
  -targets darwin-launchd-pkg    Target platforms to build. Specified in the form platform-init-package
  -transport                     Transport for launcher. Expected as grpc, jsonrpc. Default is up to launcher
  -update_channel                the value that should be used when invoking the launcher's --update_channel flag
  -with_initial_runner false     Run differential queries from config ahead of scheduled interval.
```

Targets

Binary Format	Init System	Package Format
darwin	launchd	pkg
windows	service	msi
linux	none	deb
	systemd	rpm
	init	tar
	upstart	

Targets

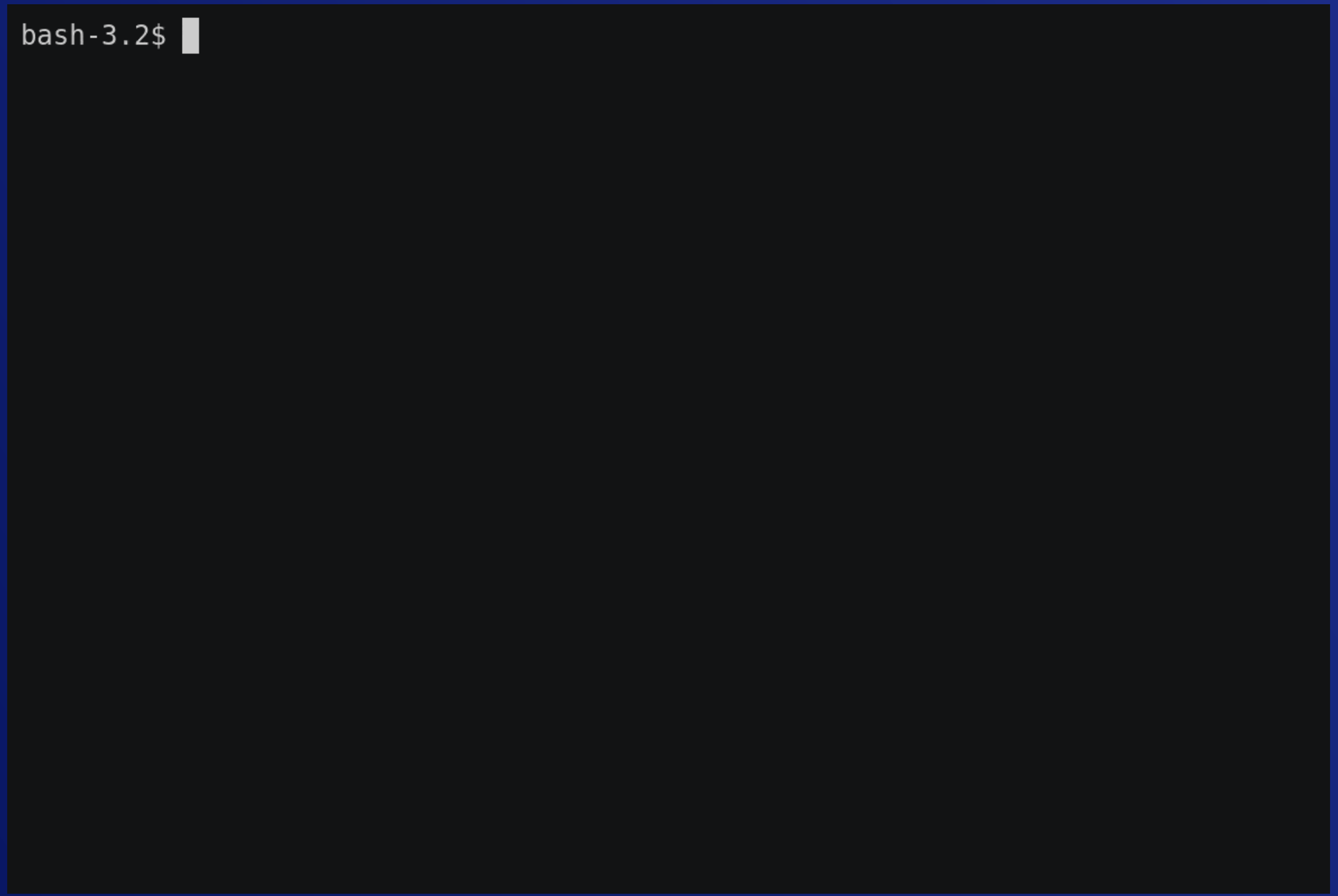

Binary Format
Init System
Package Format
darwin
launchd
pkg
windows
service
msi
linux
none
deb
systemd
rpm
init
tar
upstart

Let’s see it

KOLIDE

Make

Explore



Lessons I learned

Minimize Needless Builds

- Seems obvious in retrospect
- Don’t build ended trials
- Don’t build if there are no changes

Windows

- Similar, but also totally different
- Services are a strange beast
- Auto-Update is harder
  - No exec call
  - Cannot replace a running binary
  - WiX Toolset doesn’t expose all the service options for restart

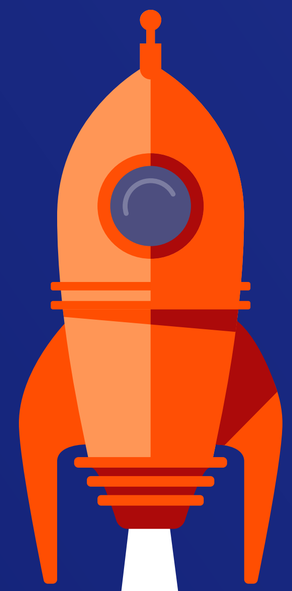
Code Signing

- Certificate management surprisingly gnarly
- Certificate stores are not always accessible how you think
- Windows is now reputation based
  - except EV certs
  - But EV certs are shipped on hardware HSMs (usually)

Go is great

- Great support for MVPs
- Allows small starts and fast iteration
- Great error handling as compared to bash
- Good test framework
- If it compiles, ship it

Questions?



KOLIDE

Useful Links

- http://kolide.com/
- https://github.com/kolide/launcher/
- https://github.com/kolide/launcher/blob/master/docs/package-builder.md
- https://github.com/kolide/fleet

Thank You!

 **seph** @ kolide.co

 github.com / **directionless**

 **seph** @ osquery Slack

 twitter.com / **twseph**