ActiveState®

Build Engineering

The Evolution of Build Engineering in Managing Open Source

Panelists

- **Pete Garcin**, Senior Product Manager, ActiveState
- **Shaun Lowry**, Build Engineering Lead, ActiveState

ActiveState®

Build Engineering and its Role in Managing Open Source

Pete Garcin, ActiveState

Build Engineering Evolution in Managing OS

Pete Garcin
Senior Product Manager
ActiveState

ActiveState®

Build Engineering Evolution in Managing OS

Capital One · Bank of America · American National · HSBC · TESCO Bank · Credit Suisse

Bombardier · GE Aviation · Northrop Grumman · NASA · Siemens · Cisco

intel · ca technologies · IBM · Microsoft · Hewlett Packard Enterprise · verizon

Track-record: 97% of Fortune 1000, 20+ years open source

Polyglot: 5 languages - Python, Perl, Tcl, Go, Ruby

Runtime Focus: concept to development to production

ActiveState

Build Engineering
Evolution in Managing OS

Open Source in the Enterprise

90%	Applications built with Open Source
76.5%	Have Vulnerabilities
54%	Not Compliant

ActiveState®

Open Source in the Enterprise

More sources, more attack surface, more repositories to manage

Build Engineering Defined

Who? Developers with:

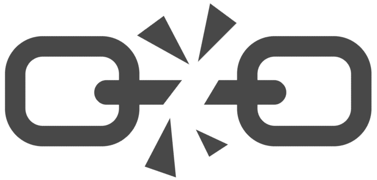
- Ecosystem knowledge
- Systems knowledge
- Process knowledge

What? Core tasks are:

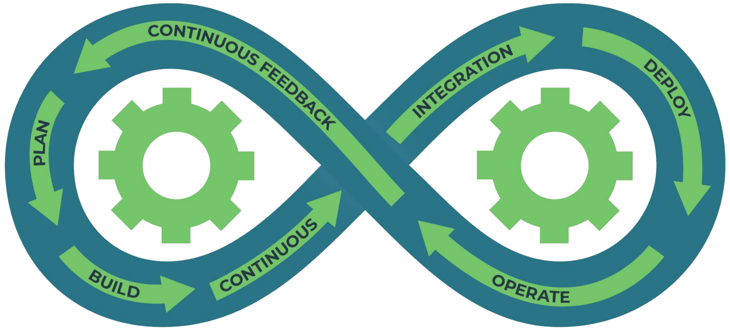
- Locating canonical sources of libraries, languages, tools, etc.
- Compiling those sources into artifacts.
- Packaging those artifacts for distribution.

Build Engineering in Relation to SDLC

Build Engineer



Missing Link

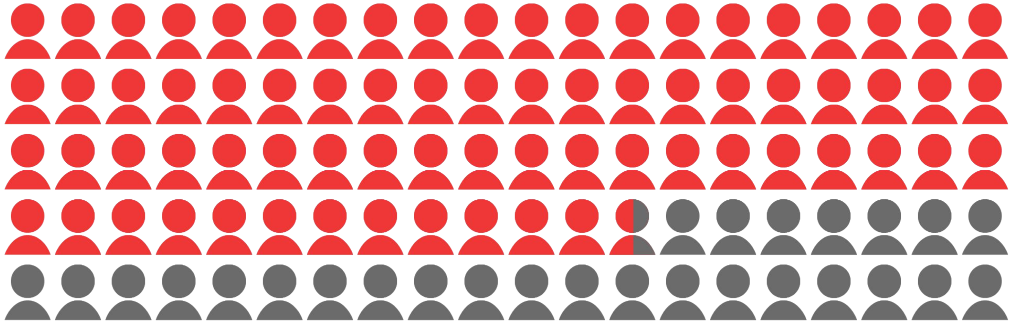


SDLC

ActiveState®

Build Engineering
Evolution in Managing OS

Hidden Costs



75%

Managing dependencies

ActiveState®

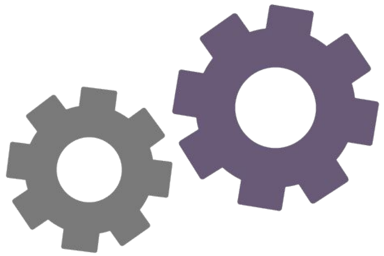
Automating Build Engineering

- Environment configuration
- Dependency management
- Build execution and storage.

What are the challenges?

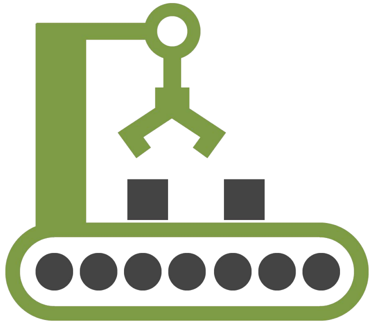
- Automate, systematize, componentize builds
- Seamless, effortless and reproducible across your team and organization.
- Reproducibility, critical for testing, deployment and development — without it, nobody is speaking the same language.

Challenges in Build Engineering



Environment
Configuration

Dependency
Management



Build
Reproducibility

ActiveState®

Environment Configuration

Toolchain
setup

Language /Tool
versions Setup

Compiler
version

SAY NO TO DEPENDENCY HELL

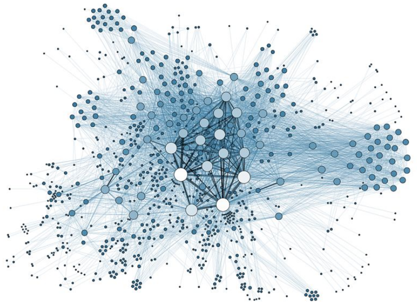
Dependency Management

Version Pinning

Monitoring CVEs, Licenses

01001

Binary Sources



Dependency Chains

ActiveState®

Automation Wins

DEBT

Developer
Time

Developer
Sanity

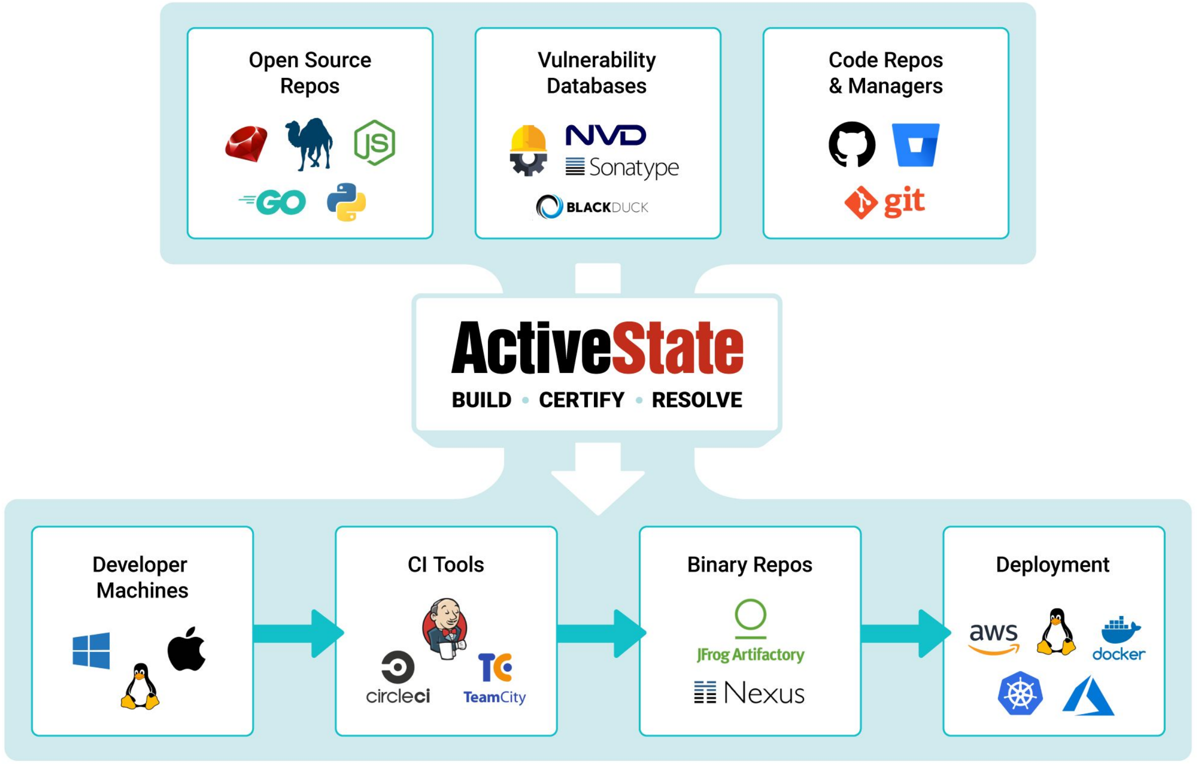
Shrink
Tech Debt

Potential Features + Automation

- “Free” speculative builds
- Build revisions as source control — can be forked, reverted, merged, etc.
- Integration with your CI
- No more local hacks, “Franken-builds”, etc. — everything is audited and guaranteed

Build Engineering
Evolution in Managing OS

ActiveState Platform



ActiveState®

Build Engineering - Why It’s Hard and Why You Still Need It

Shaun Lowry, ActiveState

Build Engineering Evolution in Managing OS

Shaun Lowry
Build Engineering Lead
ActiveState

ActiveState®

Third-party software

- Many benefits of using open source components
- OSS or licensed components offer shortcuts and competitive advantage
- How to incorporate 3rd-party components

What's so hard?

Compiling disparate OSS components

- Not all authors use the same tools
- Not all authors care about your platform(s)
- Authors might be great SMEs, but not great engineers

Compilers

- Different compilers disallow different code
- Same for compiler versions
- Some even have different ABIs between versions (GCC4 vs GCC5)

Build Tools

Autotools

Bazel

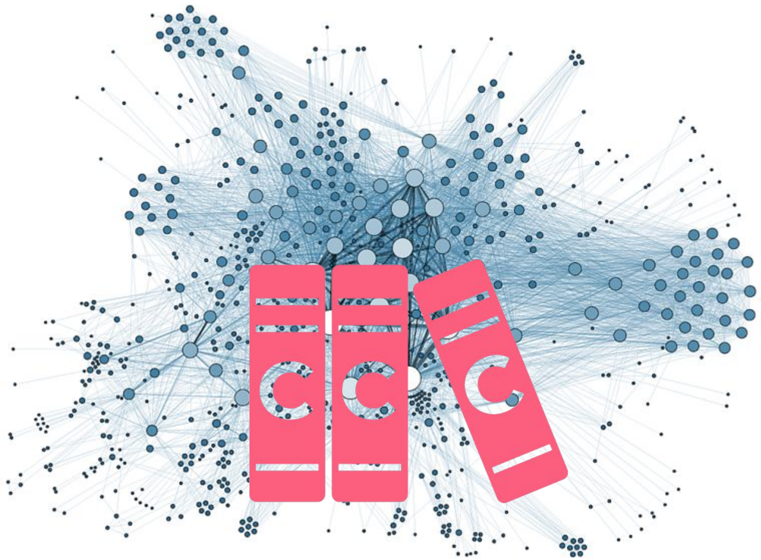
Cmake

setup.py

Custom

Build Engineering
Evolution in Managing OS

Dependencies



ActiveState®

C libraries

- **Assumed to be on the system**
  - Sometimes incompatible (e.g. libffi, libgdbm)
- **Source included with package**
  - Do not keep pace with security updates
- **Download source at build time (!)**

Example

- **Builds using bazel**
    - Tensorflow versions sensitive to bazel version
    - Take hours
- **Many variations, optional support**
    - for additional instruction sets (SSE, AVX, AVX2)
    - for GPU acceleration (CUDA)

Why?

That's scary! What's wrong with just downloading free stuff?

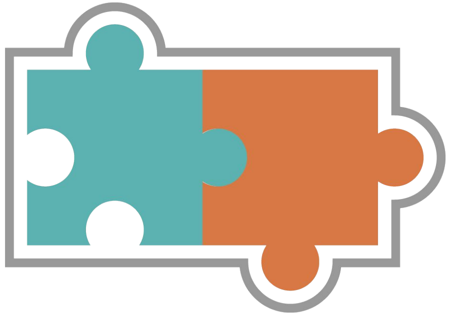
TRUST

COMPATIBILITY

LIABILITY

TRUST

- Was that zipfile/wheel/magic binary really compiled from that source?
- What else is in there?
- How many people are you trusting?
- Do your customers trust all of them?



COMPATIBILITY

- **OS or runtime dependencies**
  - Glibc, ucrt, msvcrt etc.
- **Uniform compiler**
- **Hardware dependencies**
  - Instruction set optimisations
  - GPU availability

LIABILITY

Bugs

- What happens when one hits a customer?
- How quickly can you address it?

Licenses

- Are your licenses compatible with every 3rd-party package?
- What about the packages they bring with them?

Example: wand

- **Python binding to ImageMagick**
  - ImageMagick has different licenses at different versions
- **ImageMagick needs Ghostscript for PDF manipulation**
  - Ghostscript is AGPL

So what should I do?

- Building OSS from source can be hard to deal with
- Not building from source can be worse
- Incorporate OSS builds into your own pipeline
- Outsource OSS builds to a single trusted source

Q & A

ActiveState®

What’s Next

- Watch a demo: https://www.youtube.com/watch?v=c5AIxN9ehrI
- Get a demo marketing@activestate.com
- Contact us for the language build you need: platform@activestate.com

ActiveState®

Where to find us

Tel: **1.866.631.4581**

Website: **www.activestate.com**

Twitter: **@activestate**

Facebook: **/activestatesoftware**

ActiveState®