Industrializing the Software Build Process

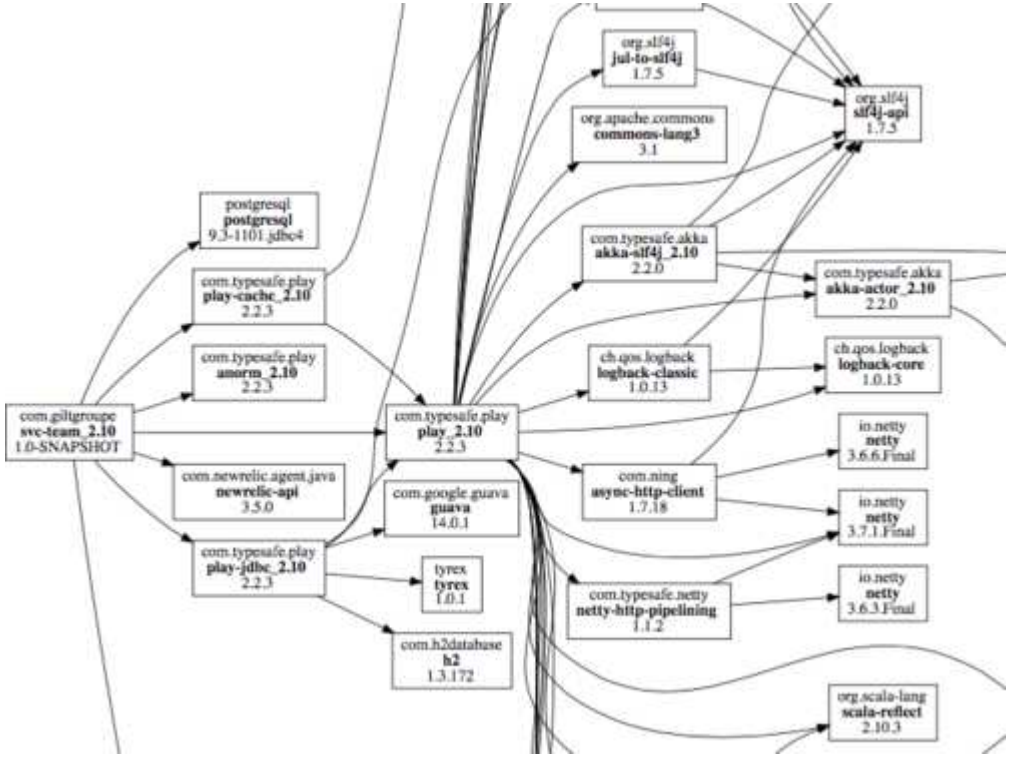
Paul Kerchen
June 2016

Agenda

- Goals of Build Industrialization
- Day-to-day Challenges
- Learning from Other Domains
- EB's Approach
- Simple Live Demonstration
- Conclusion

Introduction

- Software modules and components increase build configuration complexity
- Build configurations must evolve to keep up with changes in tools, libraries, etc.
- Most software engineers aren't interested in build configuration



Build Industrialization Goals

- Ability to make a release at any time
- Confidence in the build's integrity and quality
- Consistency of build outputs
- Harmonizes with configuration management needs

- All of these goals hinge on build configurations

Build Configurations

- Build Configurations should:
  - Be easily reproducible
  - Allow meaningful versioning
  - Produce identical build outputs
  - Allow environment and tool customizations that don't affect build outputs

- Build Configuration changes should be rolled out:
  - Consistently
  - Universally
  - Simultaneously

- No Sacred Build Machines!

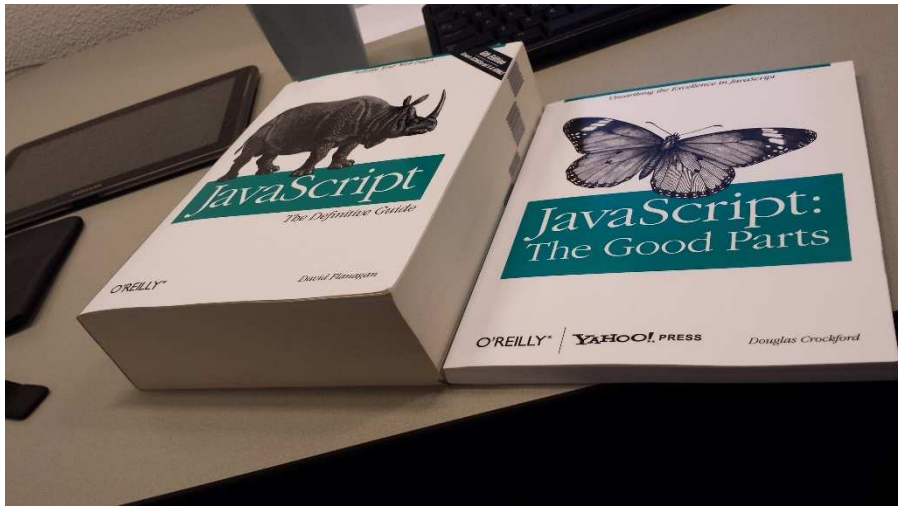
Symptoms of a Sacred Build Machine

- Only one machine capable of making production builds
- Build configuration hard to reproduce
- Different instances of build environment do not produce identical output from same input
- Aversion to change for fear of losing ‘working’ configuration

Traditional Approaches and Their Shortcomings

- Static build configuration—can hurt security and competitiveness
- Shared virtual machine (VM) images—configuration drift
- Full commercial development environments—vendor lock-in

- How to just take the good parts?



Web Development

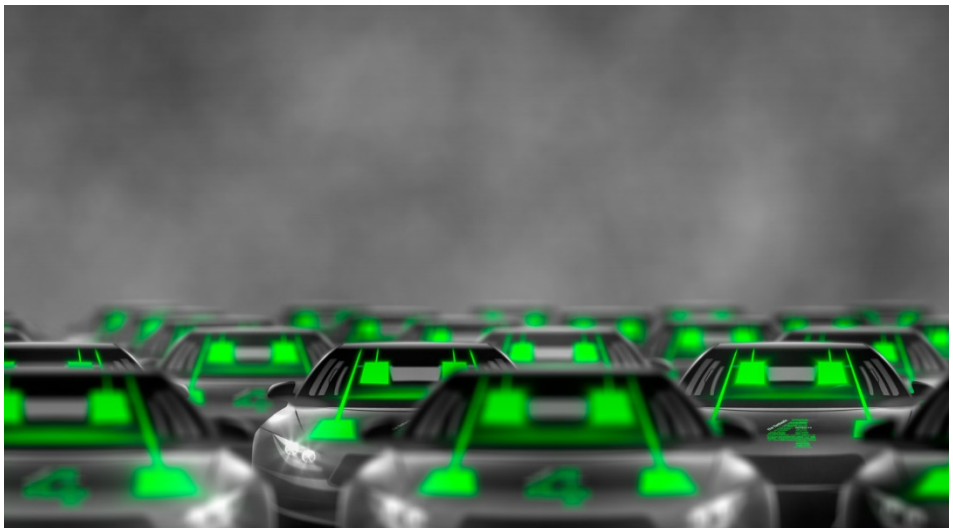
- Everything changes all the time
- Distributed development teams
- Tools and environments constantly evolving
- Applications are entire virtual machines (VMs)
- Consistent, identical VM instances are routinely rolled out

Web Development: The Good Parts

- Use VMs that are easy to create and maintain
- Minimize hard-to-version bits
- Use open source tools to save money and reduce vendor lock-in
- Embrace continuous integration (CI)

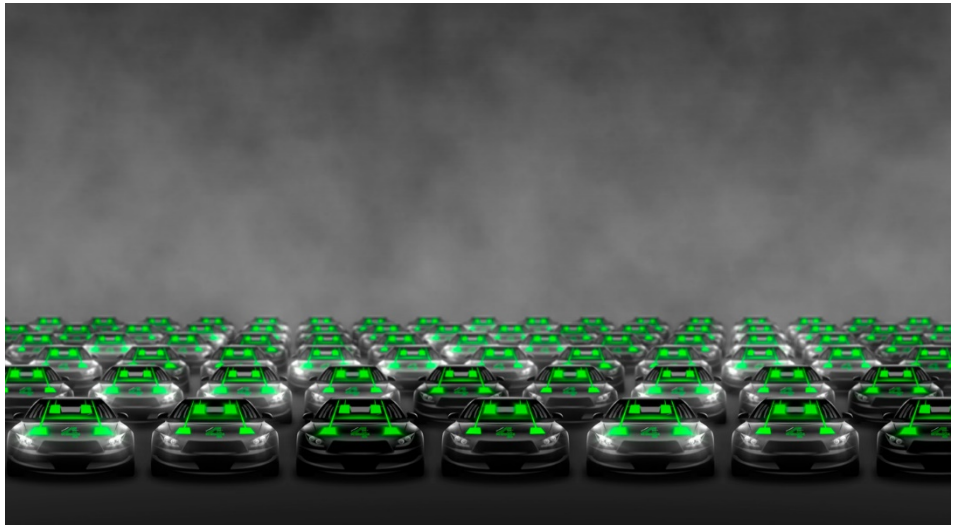
VAGRANT by HashiCorp

- Open source project with commercial additions (for VMWare)
- Creates lightweight, reproducible, and portable development environments
- Configuration in plain text files: easy to version and see changes
- Works with VMWare and VirtualBox
- Can be run headless



Packer by HashiCorp

- Integrates with Vagrant; same developer = better integration
- Explicitly designed to keep build environments consistent between development and production
- Pre-bakes VM images that can be hosted wherever convenient



A Simple Demonstration

Using Vagrant with Jenkins

- Vagrant VMs run as Jenkins slaves
- Jenkins brings VMs online as needed
- VMs used by Jenkins are the same as those used by developers

VM Updates

- Build configuration changes should be applied:
  - Consistently
  - Universally
  - Simultaneously
- Vagrant helps with the first
- The other two can be done by Chef or Puppet

Chef and Puppet

- Automates updating of VMs
- Centralized server acts as distribution hub for updates
- Server can be self-hosted locally or in the cloud (AWS or Azure Marketplace)
- Commercial hosting options also available

Summary

- Industrialized build processes increase innovation and quality, decrease time-to-market and risk
- EB can help industrialize your build processes

More Information

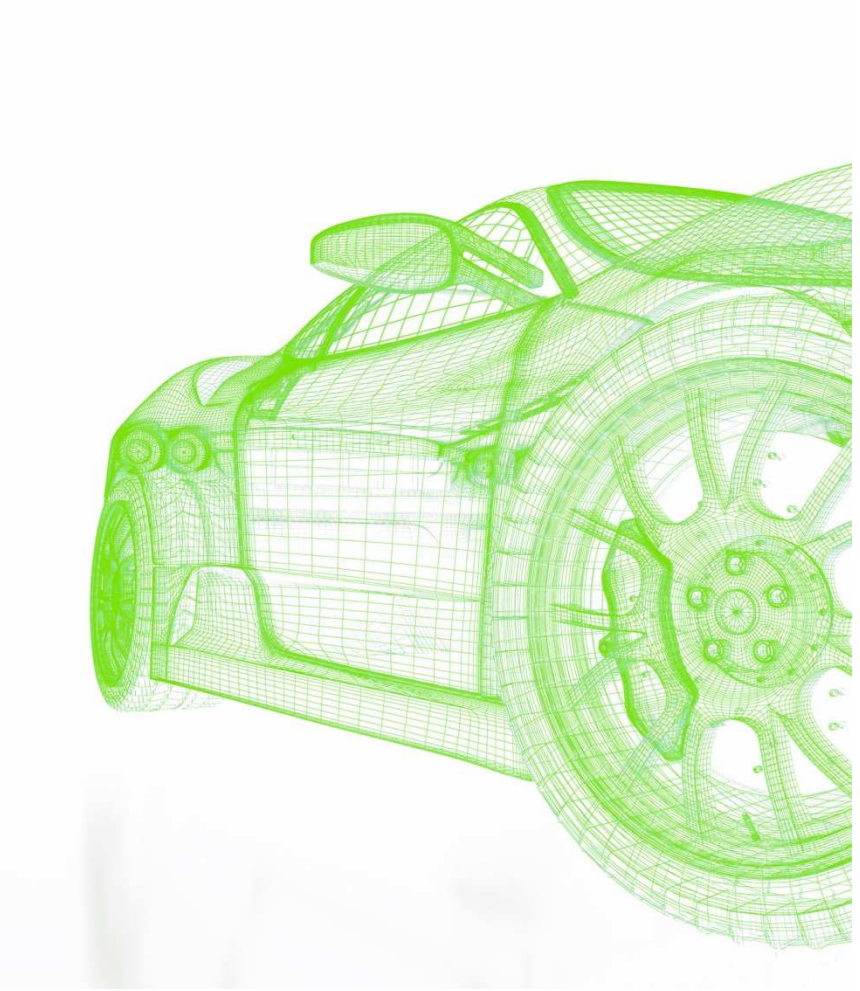
Chef
https://www.chef.io/

Jenkins
https://jenkins.io/

Packer
https://packer.io/

Puppet
https://puppet.com/

Vagrant
https://www.vagrantup.com/



Get in touch!

Contact us:
https://www.elektrobit.com
Paul.Kerchen@elektrobit.com