Lessons Learned from the
International Workshop on
Release Engineering

Bram Adams, MCIS
Stephany Bellomo, SEI
Foutse Khomh, SWAT

http://releng.polymtl.ca
PART I: RELENG?!
Back in 2009 …
On average we deploy new code *fifty* times a day.
Continuous Delivery: the Early Days

continuous integration

VCS

15k tests

test

staging/production

9 min.

6 min.

http://goo.gl/qPT6
Nowadays …
feature-based releases
(i.e., when it's done)

vs.

regular, time-based releases

openstack™
6 months

6 weeks

2 weeks (mobile)
twice/day (web)
dozens of times/day

NETFLIX
Cycle Time != Development Time
But why all this trouble?
key goal of continuous deployment is to **reduce the risk** of releasing software if it hurts, do it more frequently, and bring the pain forward

Jez Humble

Build a little and then test it. Build some more and test some more.
Before & After

• How quickly can we ship a chemspill release?
  • 4-6 weeks  11 hours
• How long to ship a “new feature” release?
  • 12-18 months  6 weeks
• How many active code lines?
  • 1 1/2  42
• How many checkins per day?
  • ~15 per day  325 per day

... aim to ship high quality products without regressing key quality metrics

Chuck Rossi
OK, what do I need for this?
Release Engineering Pipeline (2)

4. artifact repo

5. more stages of tests, acceptance tests, performance tests, UI tests, manual tests, etc.

6. deployment in production environment

7. infrastructure repo

8. release to users

---

- Cucumber
- Selenium
- jbehave
- Splunk
- Nagios
- Chef
- VAGRANT
- Puppet

- XL Deploy
- Continuous Delivery
Infrastructure-as-Code

# Install PostgreSQL server and client
include_recipe "postgresql::server"
include_recipe "postgresql::client"

# Make postgresql_database resource available
include_recipe "database::postgresql"

# Create database for Rails app
db = node["practicingruby"]["database"]
postgresql_database db["name"] do
  connection(
    :host => db["host"],
    :port => node["postgresql"]["config"]["port"],
    :username => db["username"],
    :password => db["password"],
  )
end
PERIODIC TABLE OF DEVOPS TOOLS (V1)

Share

Embed

Become Excellent!

Subscribe here!

Example Pipeline

Another Example Pipeline

Sounds easy, right?
Rapid-release Firefox meets corporate backlash

New Firefox versions now will emerge every six weeks as Mozilla tries to be more competitive. That's too rapid for some, but Mozilla is willing to leave them behind.

by Stephen Shankland @stshank / June 27, 2011 3:33 AM PDT

Last week, Mozilla programmers and executives were jubilant when the release of Firefox 5 marked the successful transition to a more competitive rapid-release development cycle.

Now, with a backlash from corporations and others who aren't equipped to handle that pace of change, things aren't quite so sunny. The organization and its community of supporters have begun some soul-searching about how to reconcile the conflicting priorities—developing software quickly but not leaving users behind.

Mozilla has concluded that Firefox isn't for corporations whose Web use doesn't move at the speed of today's Web, though. That decision frees
Latest Windows 10 update shows how rapid releases work in practice
Problems revealed by the fast track were fixed before the rollout to the slow track.

by Peter Bright - Nov 26, 2014 2:05am CET

Windows 10’s updates and maintenance are following a different, better path to all prior Windows releases: one with more regular updates and quicker access to new features for those who want it, while still offering enterprises a slower pace of delivery. With the first update to the Windows 10 Technical Preview a month ago. Microsoft also enabled the two-speed insider program for the million or so users who put Windows in their computers’ test bed.

With the three announced stable release cadences (a fast consumer-oriented one, a medium business-oriented one, and a slow critical system-oriented one), and two-speed insider program, this will leave Windows users on up to five different versions of Windows 10.
Continuous delivery for web apps is a solved problem ...

Chuck Rossi
Google, Amazon and Facebook all are using very aggressive Continuous Delivery workflows and have been doing so for years.

UPDATED January 21, 2014

I meant the adjective “mainstream” in the sense of “not dangerous.” For example: “The Ramones are so mainstream, I only listen to Norwegian Death Metal.” Based on the large amount of feedback I have received, this is not everyone’s default definition of “mainstream.” Hopefully adding this paragraph to the post will clear up such ambiguity for future readers.

Once again, I did not mean to imply that everyone is doing CD. Everyone is not doing CD! But, **CD is no longer the risky experiment it was in 2010** when Chad Dickerson hired me to help [scale the CI system at Etsy](http://infiniteundo.com/post/71540519157/continuous-delivery-is-mainstream) (which is how I got involved in this whole discussion in the first place). Today CD is a mature option and I think it is the best option available. But there are certainly other ways to build software and lots of people use
For every successful organization engaging in continuous delivery, there are many, many others that are struggling. **Difficulties include buy-in from management or other teams who do not understand the value, inability to work across silos and lack of tooling (lack of support).** These are standard problems in the area of leading change.

For every successful organization engaging in continuous delivery, there are many, many others that are struggling. Difficulties include buy-in from management or other teams who do not understand the value, inability to work across silos and lack of tooling (lack of support). These are standard problems in the area of leading change.

One Size Does not fit all

..., yet continuous delivery for mobile apps is a serious challenge
Heterogeneous Hardware, Oses, ...

- generate incremental updates for each supported old version

i18n
How Poor Software Release Management Creates Security Nightmares

With aggressive software release schedules, developers may find themselves pressured into taking development shortcuts, simply to meet a looming deadline.

As a result, aggressive software releases can have a serious and unintended side-effect: a build-up of technical debt. With every software and update release, there’s the potential for new bugs and technical vulnerabilities to be uncovered.
Are These Concerns Justified?

practitioners

RELENG

researchers
1st International Workshop on Release Engineering

RELENG 2013
May 20, 2013, San Francisco, USA

Keynote: Release Engineering as a "force-multiplier"
by John O'Duinn
(Director of Release Engineering at Mozilla Corporation)

Keynote: Against All Odds – Completely Overhauling LinkedIn's Release Process
by Roman Scheiter
(Director of Engineering Services at LinkedIn)

6 practitioner talks
2 keynotes
... and 10 academic talks

2nd International Workshop on Release Engineering 2014

Release engineering deals with all activities in between regular development and delivery of a software product to the end user, i.e., integration, build, test execution, packaging and delivery of software. Although research on this topic goes back for decades, the increasing heterogeneity and variability of software products along with the recent trend to reduce the release cycle to days or even hours starts to question some of the common beliefs and practices of the field.

The RELENG workshop series aims to provide a highly interactive forum for researchers and practitioners in release engineering to: (1)...

3rd International Workshop on Release Engineering 2015

May 19, 2015, Florence, Italy

Release engineering deals with all activities in between regular development and delivery of a software product to the end user, i.e., integration, build, test execution, packaging and delivery of software. Although research on this topic goes back for decades, the increasing heterogeneity and variability of software products along with the recent trend to reduce the release cycle to days or even hours starts to question some of the common beliefs and practices of the field.

The RELENG workshop series aims to provide a highly interactive forum for researchers and practitioners in release engineering to:

- Share knowledge
- Exchange ideas
- Discuss challenges

7 papers

... and 1 roundtable article!
PART 2: Insights from the Workshop
dependency hell  

pipeline security  
education  

success & other stories  
business value
Causing unpredictable release schedules …
To Branch or not to Branch

- branches provide physical isolation
- different channels/streams of code changes and releases possible
- hard to roll-back, needs disciplined changes and merging

a multi-branching development approach

- code owners responsible for their own subsystem:
  * only backward-compatible changes are integrated
  * control features using feature flags
- no fixing of bugs after merging, instead roll back

trunk-only development

trunk-only development

a multi-branching development approach
Consolidating All Dependencies

- no shared libraries, everything rebuilt from code
- maintaining a full dependency graph of things to build
- metrics on every action for audit and analysis:
  * ratio passing tests
  * #changes per release
  * time between releases
  * #cherry-picks per release
  * time from commit to release
- features can be guarded by feature flags
Roll Your Own Release (self-service!)

- releng team develops tools to automate the release process …
- … then developers use them to roll their own releases!
- the tools are also used to channel/promote best practices
Is our Pipeline Secure?
Security of a Release Engineering Pipeline
Is the Image Deployed into Operation a Valid Image? How can we secure a release pipeline?

- Analyse a model of the pipeline to detect vulnerabilities (from design perspective)

- Restructure and remodel pipeline to remove vulnerabilities

- In practice, it’s hard to remove all vulnerabilities
Can you Learn to be a Release Engineer? If, so How?
The 10 Commandments of Release Engineering

1- Thou shalt use a source code control system
2- Thou shalt use the right tool(s) for the job
3- Thou shalt write portable and low maintenance build files

…

10- Thou shalt apply these commandments to thyself
Since 2013, Courses on Release Engineering are Being Given at:

Polytechnique Montreal
Carnegie Mellon University
NC State University
TU Delft
RWTH Aachen University
...

Sharing *Successes and Failures* is important!
Dozens of Success Stories, Failures, Lessons Learned, …

- Focus on solving company-threatening problems instead of on technology

- Talk with, and listen to, all your customers

- Show progress (measure what customers care about)

- Take baby steps, relentless baby steps (keep it simple)

- Make updates as easy as possible
Release Engineering versus and Business
Release engineering should maximize the rate at which the company can achieve its goals!
Costs/Benefits of Release Engineering …

… can depend on:

• degree of automation
• build/test performance: $/hour (cloud bill, bonuses, equity, salary, ...)
• context switches
• (in general) time spent by test, release, software, … engineers waiting for build, test, certification, … agility!
• …
PART 3: Insights from the Special Issue
RELENG workshops maintain a community with lots of insights..., why not share it via an IEEE Special Issue on Release Engineering?

Mar/Apr 2015
IEEE Special Issue on Release Engineering - I

Table of Contents

- Guest Editors Introduction Roundtable
- Continuous Delivery: Huge Benefits, but Challenges Too
- Why and How Should Open Source Projects Adopt Time-Based Releases?
- The Highways and Country Roads to Continuous Deployment
Achieving Reliable High-Frequency Releases in Cloud Environments
Release Stabilization on Linux and Chrome
Rapid Releases and Patch Backouts: A Software Analytics Approach
Vroom: Faster Build Processes for Java
A Roundtable with Three Release Engineers

Participants:
- Mozilla (Kim Moir)
- Google (Boris Debic)
- Facebook (Chuck Rossi)

Question topics:
- What metrics are most valuable to release engineers?
- How do you maintain quality and stability?
- Are there limitations for continuous delivery?
- How can we educate others about the value of release engineering?
My Favorite Roundtable Quotes

“On a given day we might let 5% of the browser population get a new release. We have automatic crash reporting in the browser.”
Moir, Mozilla

“Mobile deployments are more challenging than Web deployments because we don’t own the ecosystem.”
Rossi, Facebook

“I tell people that release engineering is the difference between manufacturing software in a startup compared to a repeatable and predictable approach.” Debic, Google
International Workshop on Continuous Software Evolution and Delivery (CSED)

Important Dates
• Abstract: January 15, 2016 AoE (mandatory)
• Submission: January 22, 2016 AoE
• Notification: February 19, 2016
• Camera Ready: February 26, 2016
• Workshop: May 14-15, 2016

Stay tuned for RELENG 2016!
"Thank You"
CALL AGAIN
and bring
a Friend!
feature-based releases (i.e., when it's done) vs. regular, time-based releases

openstack 6 months
Google 6 weeks
Facebook 2 weeks (mobile)
Netflix twice/day (web)
Facebook dozens of times/day

dependency hell
dependency hell
pipeline security
pipeline security
education
education

A Roundtable with Three Release Engineers

Participants:
- Mozilla (Kim Moir)
- Google (Boris Debic)
- Facebook (Chuck Rossi)

Question topics:
- What metrics are most valuable to release engineers?
- How do you maintain quality and stability?
- Are there limitations for continuous delivery?
- How can we educate others about the value of release engineering?