The 10 Commandments of Release Engineering

Dinah McNutt
Google, Inc.
mcnutt@google.com

Google™

Release Engineering

"Accelerating the path from development to operations"

Google™

Overview

- These commandments are truths based on my 20+ years of developing commercial software
- Concepts apply to software products for both internal and external customers
- Ideas presented are my own, not necessarily Google's

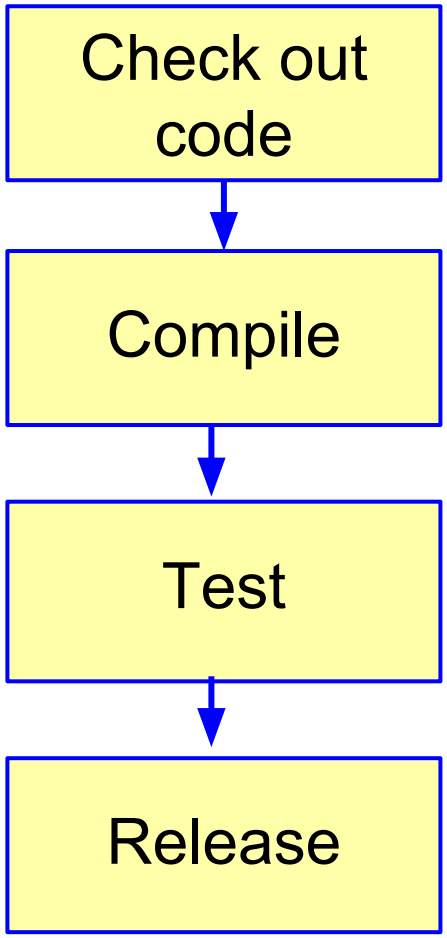
Google™

Background

- Release processes are usually an afterthought
- Most systems do the minimum required to "get it done"
- Release processes should be treated as products in their own right
- There is often a big disconnect between the developer writing the code, the person writing tests, and the system admin who installs it

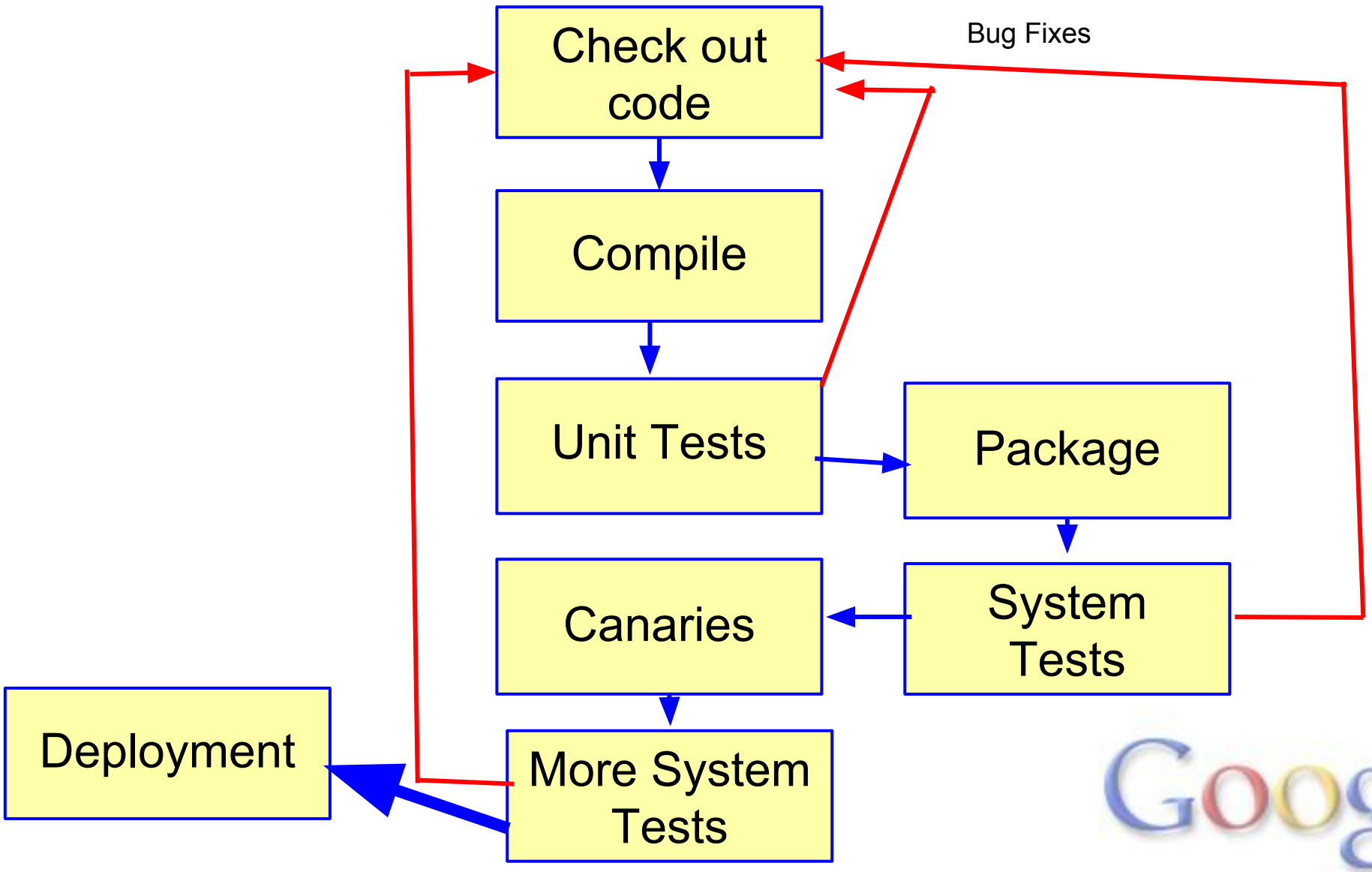
Google™

Steps in Release Process

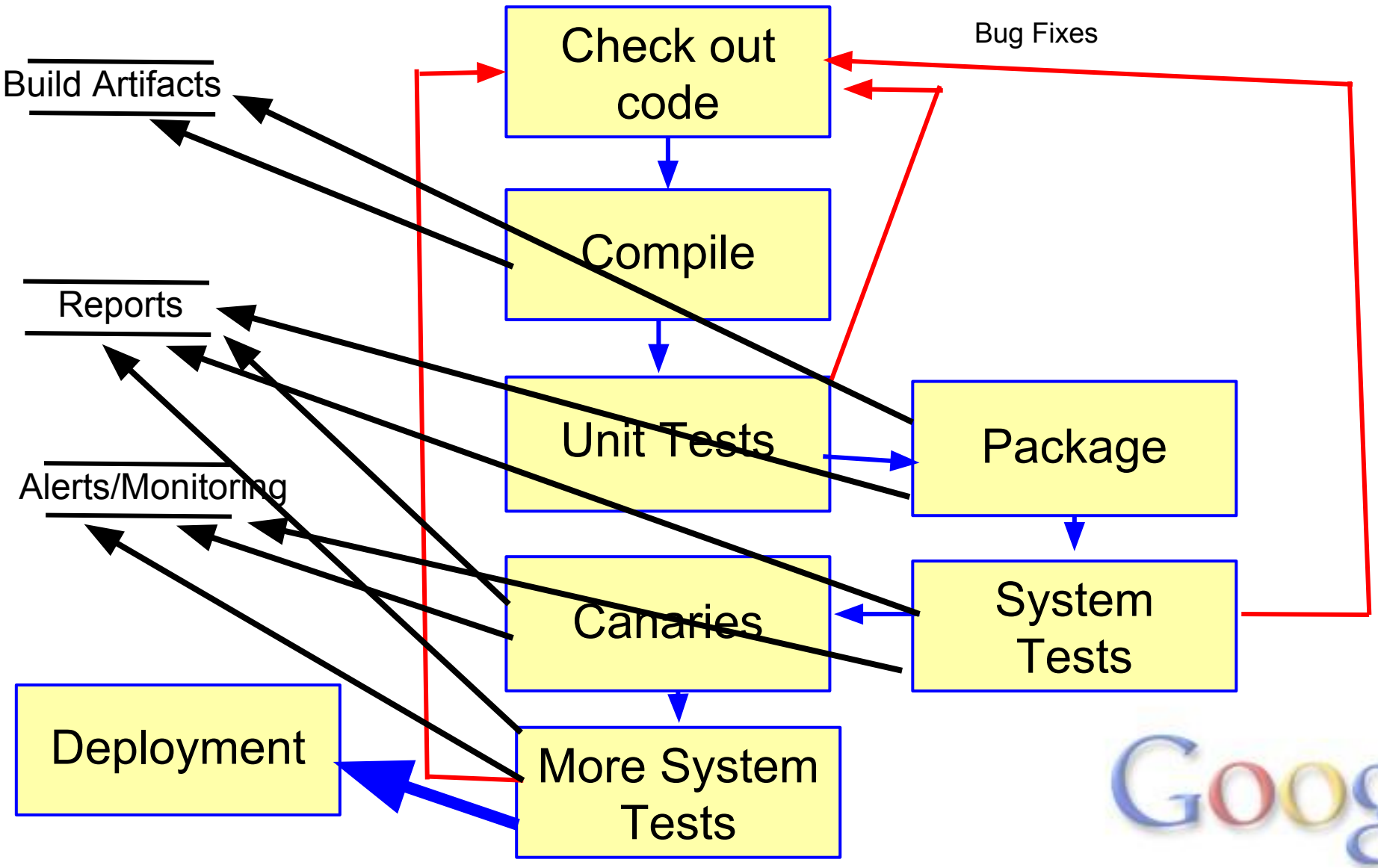


Google™

The Real Process



The Real, Real Process



I

Thou shalt use a source code control system

Google™

I - Thou shalt use a source code control system

- **Everything** needed to release should be under source control
  - source code
  - build files
  - build tools
  - documentation
- Doesn't matter what you use, just use something!

Google™

Reproducible Build Environment

- Not usually checked into a SCR, but still may need to be recreated:
  - Operating System
  - Compilers
  - Build tools
- Possible solutions:
  - Backups
  - Installation servers
  - Virtual machines

Google™

Configuration Management

- Binary dependencies
- Configuration files
- Manifests
- Change lists
- Machine configurations

Google™

But what about binaries?

- Binaries don’t belong in an SCM
- If you must, consider a separate repository

Google™

Moral

Reproducibility is a virtue

Google™

II

Thou shalt use the right tool(s) for the job

Google™

II - Thou shalt use the right tool(s) for the job

Complex projects may require multiple build tools

Examples:

- `ant, maven, gradle` for java
- `make` for C and C++ - the dependency checking is crucial
- scripting languages (bash, python, etc.)

Google™

Moral

Unnecessary complexity is a sin

Google™

III
Thou shalt write portable and low-maintenance build files

Google™

III - Thou shalt write portable and low-maintenance build files

- Plan to support multiple architectures and OS versions
- Use centralized configuration files for definitions common to build files
  - Compiler options will change between architectures
  - Editing hundreds of files for a single change is no fun
- Provide template files so developers can easily create new build files

Google™

Use a unique build ID

- Must provide enough information so the build can be uniquely identified and reproduced

- Examples:
  - 164532_20131008_2_RC00
  - 20131008_RC05

- Must be easily obtainable
  - Included in packaging
- Embedded in binaries

Google™

Moral

Measure twice, cut once.

Knowing your ancestry is a virtue.

Google™

IV
Thou shalt use a release process that is reproducible

Google™

IV - Thou shalt use a release process that is reproducible

And automated...
And unattended...
And reproducible...

- Adopt a continuous build policy
- Leverage open source tools like Jenkins, bamboo, buildbot, teamcity
- Write your own

Google™

Release Engineering as a Service

- Developers should be able to operate in self-service mode
- Tools should support implementation of best practices and policies
- This is where release engineers can offer significant value to their organizations

Google™

V

Thou shalt use a package manager

Google™

V - Thou shalt use a package manager

- Auditing
- Leverage installation/upgrade/removal capabilities
- Package summary (who, what, when, etc.)
- Built-in version tracking and dependency checking
- Manifest
- Use native package managers when possible

Google™

Moral

tar is not a package manager...

Google™

VI
Thou shalt design an upgrade process before releasing version 1.0

Google™

VI - Thou shalt design an upgrade process before releasing version 1.0

- Packaging decisions can affect the ability to upgrade
- Design an upgrade process at the same time you are designing an installation process

Google™

Provide a complete install/upgrade/uninstall process

- Totally automated process
- Rollback AND roll forward
- Packages should be relocatable

Google™

Moral

Not thinking ahead is a sin.

Google™

VII
Thou shalt provide a detailed log of what thou hath done

Google™

VII - Thou shalt provide a detailed log of what thou hath done

- Installing/Patching/Upgrading/Removing software should provide a detailed log of what is happening
- Provide the ability to unpack and inspect the packages without installing
- Ideally there should be a "do nothing" option so I can see what is going to happen first
- Critical for troubleshooting problems

Google™

VIII
Thou shalt canary

Google™

VIII - Thou shalt canary

- Practice of using domestic canaries to detect carbon monoxide in coal mines
- In software, refers to rolling out a release to a small number of users
- Many problems only show up in a production environment
- Canarying can allow early detection

Google™

IX
Thou shalt keep the big picture in mind

Google™

IX - Thou shalt keep the big picture in mind

- Remember, it’s Dev -> DevOps

Google™

X

Thou shalt apply these commandments to thyself

Google™

IX - Thou shalt apply these commandments to thyself

- Treat your release tools as products in their own write
- Dogfood your own best practices

Google™

Shameless Plug

The 2014 USENIX Summit on Release Engineering

June 20, 2014 Philadelphia

“From Dev to DevOps”

dinah@usenix.org

Google™

Sneak Peak at URES ‘14

- Caskey Dickson, Google, *“By their powers combined--monitoring and automated releases are like peanut butter and chocolate”*
- Chuck Rossi, Facebook,*”Moving to Mobile”*
- Daniel Cordes, Portware, *“Deploying without a Web”*
- Daniel Zapata, Netflix, *“Going from 3 week to daily releases at Netflix”*

Google™

Sneak Peak at URES ‘14

- Glenn Brown, Maven Wave Partners, *“It Works on My Machine! How Container Technologies like Docker can Revolutionize Continuous Integration”*
- J. Paul Reed, Panel Discussion on *“The Future of Release Engineering”*
- Jared Morrow, Basho, *“Building Enterprise Software on GitHub”*
- John O’Duinn, Hortonworks, *Keynote*

Google™