Introduction to Partner QE

Kernel QE, Red Hat
April 2015

Overview of Presentation *

- Welcome to Red Hat Partner QE
- Benefits
- Red Hat expectations
- Partner QE Engineer responsibilities
- What can/cannot be shared
- Beaker test system

* Based on Prarit Bhargava’s “Introduction to Partner Engineering” presentation

Welcome!

Welcome to Red Hat Partner QE

- Historically no way for Partner to easily get code into Red Hat products
  - Scheduling conflicts
  - Engineering conflicts (code reviews)
  - Hardware conflicts
- Red Hat introduced Onsite Engineer Program in 2004
- Now called “Partner Engineer Program”
- Partner QE Engineer “plugged in” the same as Partner Engineer (serves as remote QE resource)

Welcome to Red Hat Partner QE

- Partner QE Engineer
  - Schedules
  - Unreleased builds and composes
  - Internal IRC and communication
- Partner Company
  - Influences RHEL
  - Dedicated person(s)

Benefits

Benefits to Partner: Schedule Information

- Internal schedule deadlines
- Your opinion on Bugzillas and status
- Deadlines for various releases (Alpha, Beta, etc.)

Benefits to Partner: Code and Communication

- Internal email lists
- Internal IRC
- Onsite
  - As necessary

Benefits to Red Hat: You!

- Better understanding of your company’s concerns
- Better knowledge of your company’s products

Expectations

What Partner Should Expect From Red Hat

- You are like a Red Hat engineer
  - Not adversarial
- Red Hat wants you to succeed
- Assistance with testing infrastructure
  - Usage of beaker
  - Physical access to hardware

What Red Hat Should Expect From Partner

- Help acquire and support hardware in Red Hat
- Create bugs in Bugzilla using established guidelines
- Test Red Hat releases for parent company’s hardware
- Be aware of and conform to release schedules

Partner QE Engineer Responsibilities

Partner QE Engineer Responsibilities

- Bugs
  - Enter into Bugzilla: https://bugzilla.redhat.com/
  - CC or assign to Partner Engineer
  - CC Partner Manager
  - Three ACK Process

Bugzilla States

NEW*	- Nobody working on it
ASSIGNED*	- Engineer is working on it
POST*	- Patch is posted internally for review
MODIFIED	- Patch is officially committed to tree
ON_QA	- Kernel with patch is released for test
VERIFIED	- QE has verified patch
RELEASED	- Kernel with patch is released to public

* Partner Engineer can control these states

Schedule

- Default naming scheme is X.Y.Z
- Every major release has a minor release
  - Each minor release has an associated schedule
- Every minor release has update releases (also known as “z-stream” releases
  - Each z-stream release has an associated schedule
- Examples: RHEL 7.2, RHEL 6.7, RHEL 6.6.z, etc.

Minor Release QE Schedule Example

Testing Phase Begins	March 10, 2015
Alpha	April 1, 2015
Beta	April 15, 2015
Snapshot 1	May 6, 2015
Snapshot 2	May 20, 2015
Snapshot 3	June 4, 2015
Snapshot 4	June 11, 2015
Snapshot 5	June 18, 2015
Testing Phase Exit	July 13, 2015

What Can and Cannot Be Shared

Sharing Information

- Consider everything restricted information
- “Err on the side of caution”
- If you want to share anything, ask your RH manager

What cannot be shared with my company?

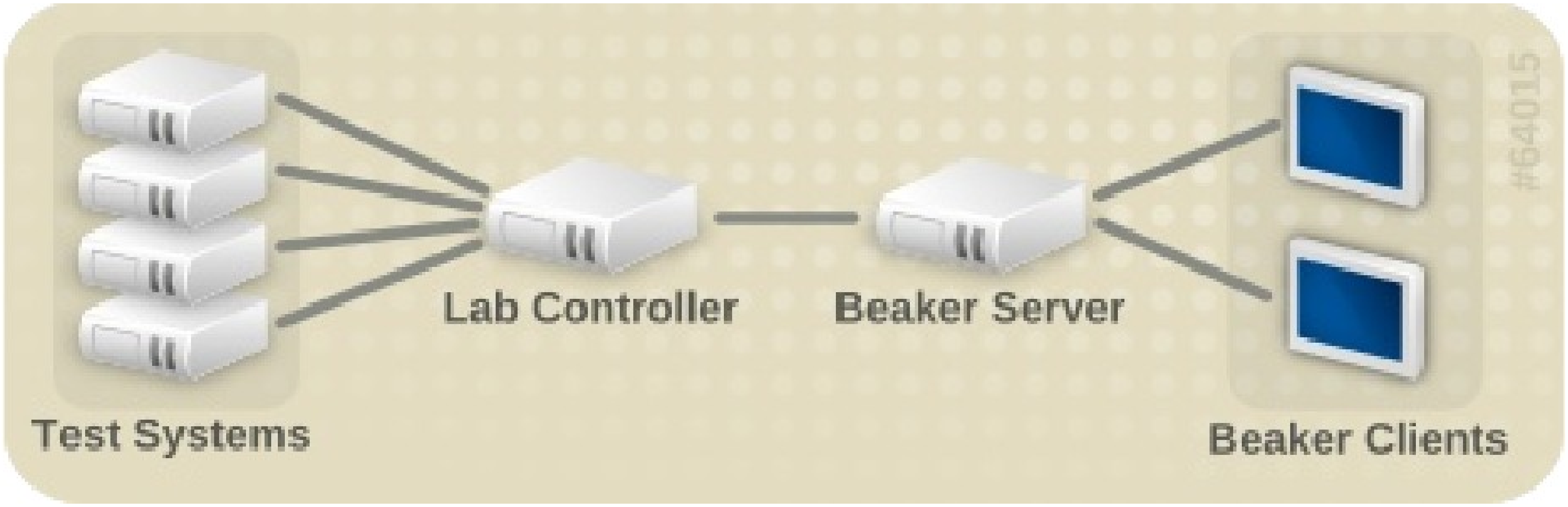
- Same restrictions as Partner Engineer

Beaker Test System

Beaker Overview

- Open-source software for managing and automating labs of test computers
- Manage systems across multiple labs
- Maintain an automated inventory of system hardware details
- Provision task execution environments on systems
- Schedule tasks to run on one or more systems
- Store and view task results

Beaker Architecture



Beaker Architecture

- Lab Controller
  - Maintains inventory and distro data
- Beaker Server
  - Central point where all job related activity takes place
  - System inventory and ability to provision systems controlled from here
  - Holds the repository of tasks

Beaker Architecture

- Beaker Client
  - Shell based Command Line Interface (CLI)
- Test Harness
  - Responsible for executing tasks on the system
  - Current harness beah

Beaker

- Benefits:
  - Provision systems
  - Create and execute automated tests across one or more systems using multiple languages
  - Large hardware inventory with easy device/system look-up capability
  - Convenient Beakerlib functions

Beaker

- Limitations:
  - ???

- Future:
  - Restraint?
  - ???

Beaker – More Information

- For more information on Beaker, go to
  - Main Beaker Project page
    - https://beaker-project.org
  - Beaker Quick Start Guide
    - https://beaker-project.org/psss-beaker-quick-start-guide-slides.pdf

END