

Xen Project Contributor Training

Part 2 : Processes and Conventions

Lars Kurth

Community Manager, Xen Project

Chairman, Xen Project Advisory Board

Director, Open Source Business Office, Citrix



[lars_kurth](#)



Content

Principles: Openness, Transparency, Meritocracy

Roles: Maintainers, Committers, Project Lead

Decision Making and Conflict Resolution

Design reviews

Bug reports

Patch contribution workflow

- Anatomy of a good patch series
- Coding style
- Personal repos hosted by Xen Project
- Staging-to-master pushgate and automated testing

Release Manager Role and Release Process

Access to Coverity Scan

Hackathons, Developer meetings, Ad-hoc meetings to resolve issue

Earning “status” in the Xen Project community



Principles and Roles:

www.xenproject.org/governance.html

Principles: Openness, Transparency, Meritocracy

Openness: The Xen Project is open to all and provides the same opportunity to all. **Everyone participates with the same rules.** There are no rules to exclude any potential contributors which include, of course, direct competitors in the marketplace.

Transparency: Project discussions, minutes, deliberations, project plans, plans for new features, and other **artifacts are open, public, and easily accessible.**

Meritocracy: The Xen Project is a meritocracy. **The more you contribute the more respect and responsibility you will earn.** Leadership roles in Xen are also merit-based and earned by peer acclaim.

Roles: Maintainers, Committers, Project Lead

- Maintainers
 - Own one or several components in the Xen Project tree
 - **Reviews and approves** changes that affect their components (Acked-by)
 - It is a maintainer's prime **responsibility to review, comment on, co-ordinate and accept patches** from other community member's.
 - It is a maintainers prime responsibility to **maintain the design cohesion of their components**. Which implies: **quality, maintainability, system properties, ...**
- Committer
 - Maintainer with **write access** to the source tree
 - Acts on the wishes of maintainers (Acked-by)
 - Allowed to act as referees should disagreements amongst maintainers arise
- Project Lead
 - Public face of project
 - Allowed to act as referee should disagreements amongst committers arise



Decision Making:

www.xenproject.org/governance.html

Decision Making: Lazy Consensus

Lazy consensus is a mutual consent decision making process between you and the community that states **your default support for a proposal is "yes", unless you explicitly say "no"**.

Restrictions:

- Any **"No"** statement must be accompanied by an explanation
- Excludes **decisions that require a sign-off**, e.g. an Acked-by a maintainer on a patch. But, if there are several maintainers that need to agree, the affected maintainers operate using Lazy Consensus
- **Excludes formal voting**, e.g. committer election, governance changes, ...

Assumptions:

- Require everyone who cares for the health of the project to **watch what is happening, as it is happening**.

Lazy Consensus: Examples

The design discussion leading to this point in time was lengthy with some disagreements amongst core developers. It took 2 months and 43 email exchanges to get to the point below.

The example shows how the proposer prompting the Maintainers for clarification on whether all the issues have been resolved and by doing so, got to an agreement.

[Contributor]

In this case, if libvirt/XenAPI is trying to query a domain's cache utilization in the system (say 2 sockets), then it will trigger `_two_` such MSR access hypercalls for CPUs in the 2 different sockets.

If you are okay with this idea, I am going to implement it.

[Maintainer]

I am okay with it, but give it a couple of days before you start so that others can voice their opinions too. Dom0 may not have a vcpu which is scheduled/schedulable on every socket.

[snip: ... there was another short exchange clarifying a question, which were addressed during the conversation]

[Contributor]

No more comments on this MSR access hypercall design now, **so I assume people are mostly okay with it?**

[Another Maintainer]

Yes -- I think everyone who commented before is satisfied with that approach, and anyone who hasn't commented has had ample opportunity to do so

Conflict Resolution : Informal

Maintainers and core developers sometimes hold different opinions regarding architecture, design or other issues. That is entirely normal. Because such situations can delay progress and turn away contributors, the project has some mechanisms to resolve this.

Informal:

- Ask the parties that disagree to resolve the disagreement
- Leave some “reasonable” time period to allow the disagreeing parties to come to a conclusion
- If in doubt, you can ask one of the committers and/or the community manager for advice.

This works in most cases.

Conflict Resolution : Formal

In situations **when no resolution can be found informally**, refereeing can be used. People with higher “status” in the community can make decisions on behalf of the community.

Example: Two **maintainers disagree on a design**. You have asked for the issue to be resolved, but no resolution could be found.

Formal:

- **Ask the referee** (in this case the committer responsible for the maintainers) to resolve the disagreement
- If in doubt, ask the community manager for advice.

Factors to Consider

- Referees do not always **proactively step in and resolve an issue**
 - Workload : may have missed a disagreement
 - Resolving issues is harder for some people than others
- Asking for an issue resolution in **public or private?**
 - Prompting disagreeing parties to resolve an issue publicly is not always easy
 - It is OK, to ask a referee or the community manager for advice privately
 - However, **Transparency** and **Lazy Consensus** require that discussions and decisions are made in public. This means that
 - **It is OK to do preparation work to resolve a conflict in private** (e.g. on IRC, a phone call, etc.)
 - But, in such a cases, there **needs to be a clear statement on the list that shows the outcome** and invites others to provide feedback

Example

A maintainer stepped in to understand and resolve an issue by having a quick conversation with one party involved: he **stated** the fact that there was a private discussion, **summarized** it and **invited others to comment**.

[Maintainer]

So XXX and I had a chat about this [the disagreement that came up in a discussion], and I think we came up with something that would be do-able. (This is from memory, so XXX please correct me if I missed anything).

So the situation, as I understand it, is:

...

That should be a good balance -- it's not quite as good as having as separate daemon, but it's a pretty good compromise.

Thoughts?



Design Reviews:

Undocumented Convention

Design Reviews

The Project has **no formal requirement** to submit designs before a patch

BUT:

- Designs are **welcome, for complex designs**
- Sometimes a fully fledged design is not necessary : a set of questions, can iteratively lead a design
- If in doubt, as to whether a design is necessary ask the community for input

Example: Question leading to a design

See [“cpufreq implementation for OMAP under xen hypervisor”](#)

[Contributor]

Hi to all.

I want to implement an cpufreq support for OMAP processors in xen. I use the Linux kernel as Dom0.

I know that there are 2 implementations of cpufreq: Domain0 based cpufreq and Hypervisor based cpufreq. But those implementations are made only for x86 architecture, not for the ARM architecture.

Could anybody give me an advice how to do that?

After an initial answer, the proposal was iteratively improved leading to a design.

The design turned out to be more complex than anticipated because of dependencies with Linux and architectural differences between x86 and ARM.

Example: Fully Fledged Design

See “[FIFO-based event channel ABI design \(draft B\)](#)”

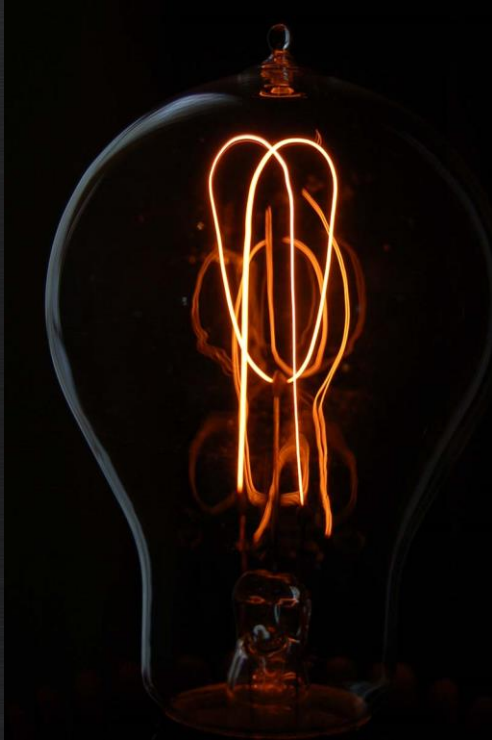
- This was an example of a fully fledged detailed design.
- Note that there was a [version A](#) beforehand – a precursor of draft B
- The design was competing with an entirely different design by Wei Liu, for which code already existed – some of which had been posted and reviewed

The community had to make a decision, which design to go for.

- It turned out that a prototype could be put together relatively quickly and an RFC followed.
- This then led to a community decision to go for the FIFO-based event channel (with the agreement of Wei Liu)

If you compare the amount of questions, these were similar to the previous example, but took [considerably less elapsed time to review](#).

This was mainly due to the fact that most developers were within one time-zone and that the design was well thought through.



Considerations:

There is no right or wrong approach

A more **iterative design review** (based on a problem or idea which is not fully defined),

- may be less predictable and
- depends on the quality of communication between proposer and reviewers

A **fully fledged design**,

- may require significant re-work if there are issues with the use-cases, wrong assumptions, etc.

Design as Documentation

Example: FIFO-based event channel ABI design

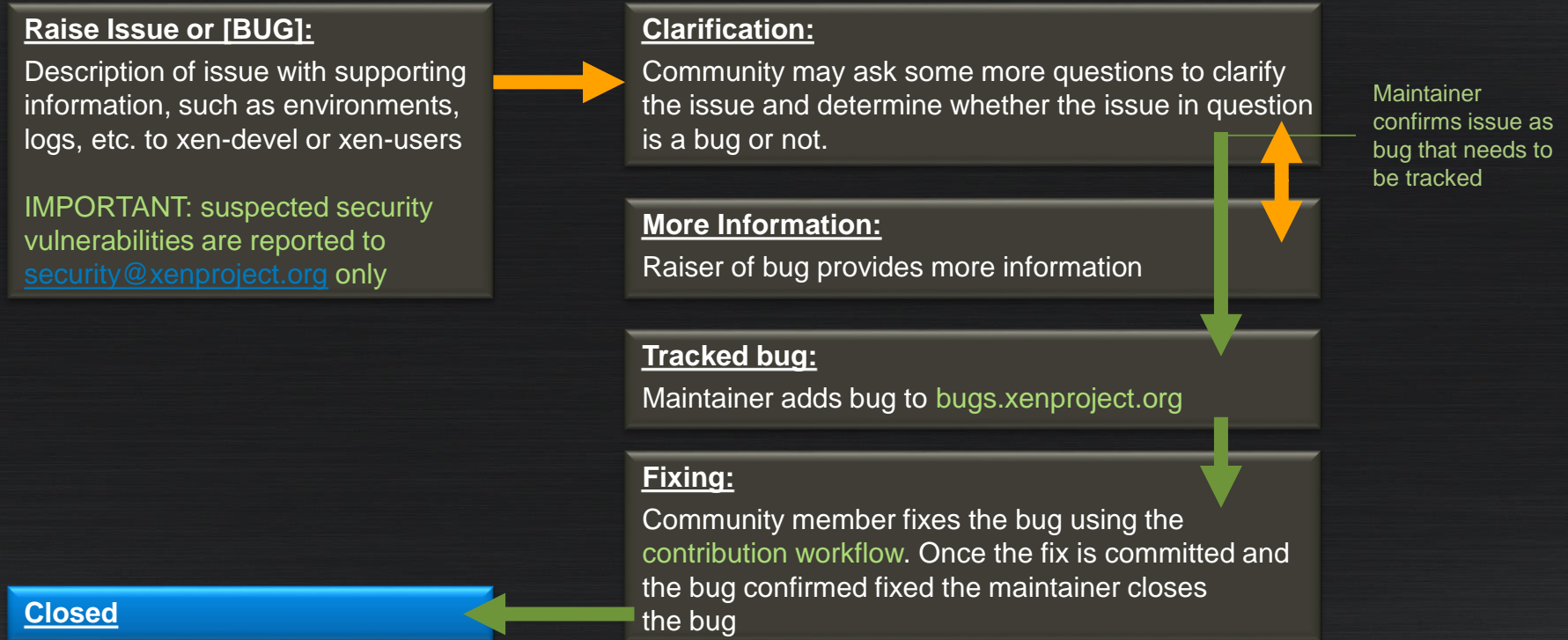
- The FIFO-based event channel ABI design had a further 6 revisions (up to draft H)
- It was kept up-to-date with the implementation. See
 - lists.xenproject.org/archives/html/xen-devel/2013-11/msg01414.html referring to
 - xenbits.xen.org/people/dvrabel/event-channels-H.pdf
- The design doc now serves as detailed documentation for the ABI.



Bug Reports:

wiki.xenproject.org/wiki/Reporting_Bugs_against_Xen_Project

E-mail based bug reporting Process





Security Vulnerabilities:

www.xenproject.org/security-policy.html

Reporting Security Bugs

Raise Security Issue:

Description of issue with supporting information, such as environments, logs, etc. security@xenproject.org only



Xen Project Security Team handles the issue

Pre-disclosure:

Members of pre-disclosure list are notified of issues and updates



Full Disclosure:

Security team announces security issue publicly at disclosure date





Security Process:

www.xenproject.org/security-policy.html

Credit for this section of the talk: George Dunlap

The “Old” Days

vulnerability
introduced



vulnerability
reported



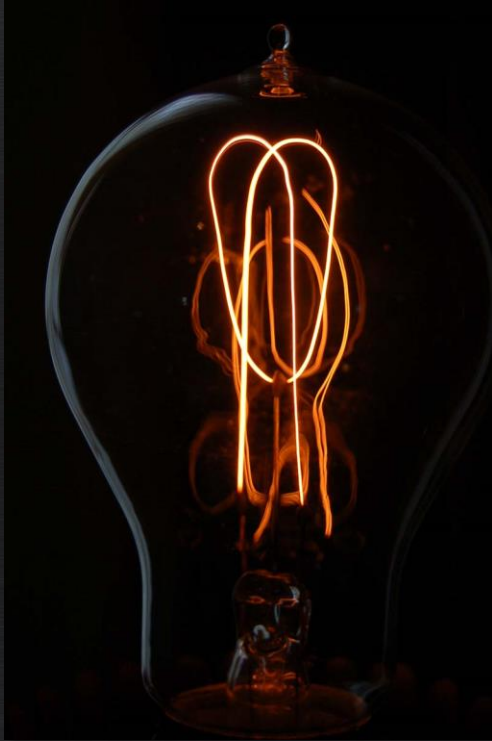
(time passes...)

vulnerability
fixed



vulnerability
fix deployed





Realization:

Vulnerability Reporters are
actually in control
(nobody can stop them from releasing information)

Full Disclosure

vulnerability
introduced



vulnerability
reported



vulnerability
fixed



vulnerability
fix deployed



Responsible Disclosure Software Providers

vulnerability introduced



vulnerability reported



vulnerability fixed



vulnerability deployed



Responsible Disclosure Service Providers

vulnerability
introduced



vulnerability
reported



vulnerability
fixed



vulnerability
fix deployed



Goals of the Security Process

- Encourage people to report bugs responsibly
- Minimize time that users are vulnerable to attack
- Maintain trust in the community

Policies

Timing of disclosure

- Honor the wishes of the reporter
- Suggest time period: 2 weeks

Pre-disclosure list

- Open to any “genuine provider” of software / service using Xen

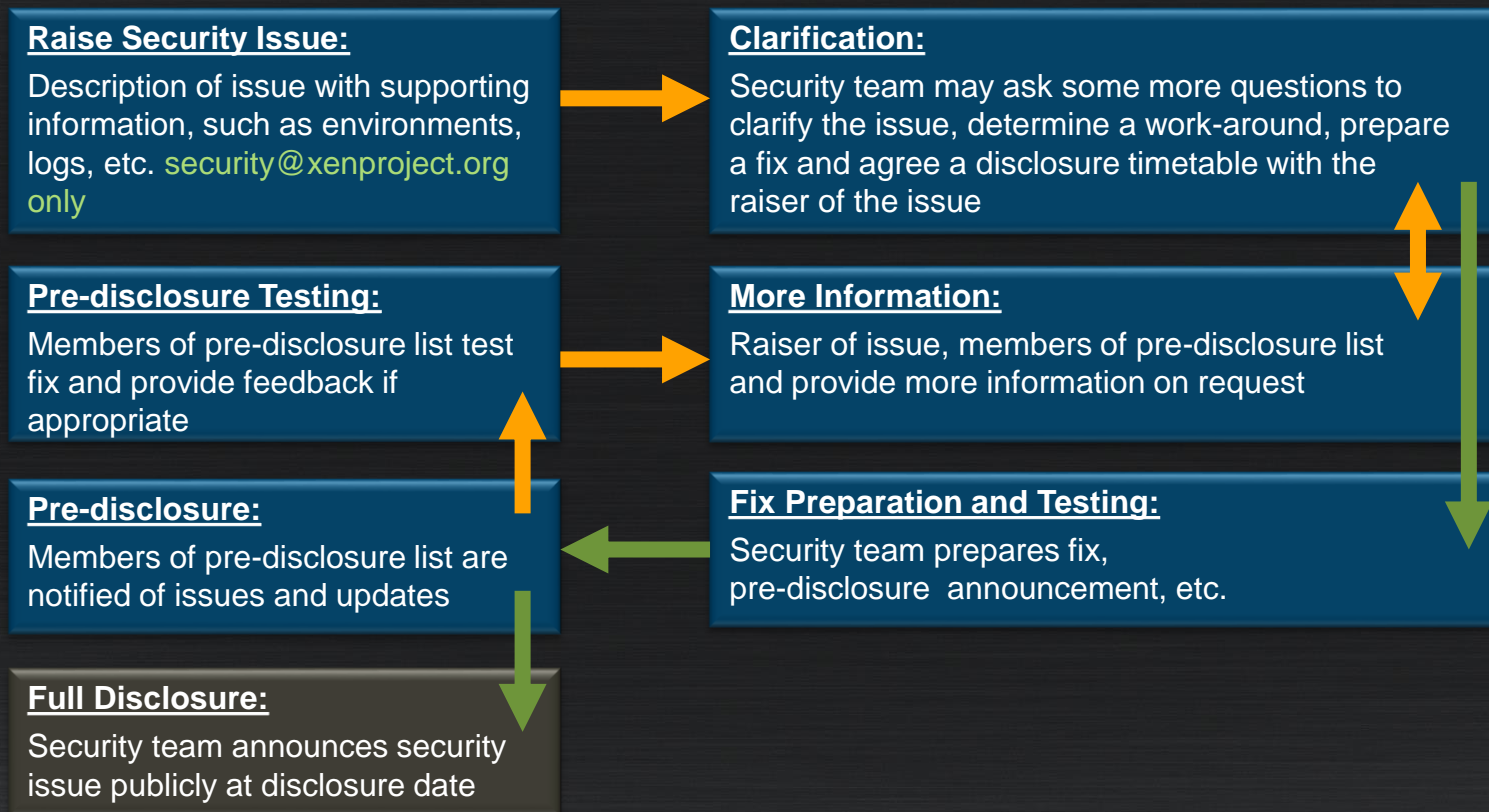
Open issues

- Software provider providing early fix to service provider (e.g. Rackspace / Softlayer) which are also on the pre-disclosure list
- Misinterpretations of text

Xen Project Security Team

- Distinguished Community Members
- Read vulnerability reports
 - Determine if it is a vulnerability
 - Come up with a fix (bringing in others if necessary)
 - Coordinate disclosure
- Manage predisclosure list according to policy

Security Bugs (revisited)



What happens if the Security Team

- ...doesn't honor the wishes of the reporter?
 - If the reporter doesn't trust the process, they may go with full disclosure
- ...favors some group in the community (aka is not impartial)?
 - Massive loss of trust in the Xen Project
 - Possible legal repercussions for anti-trust violations

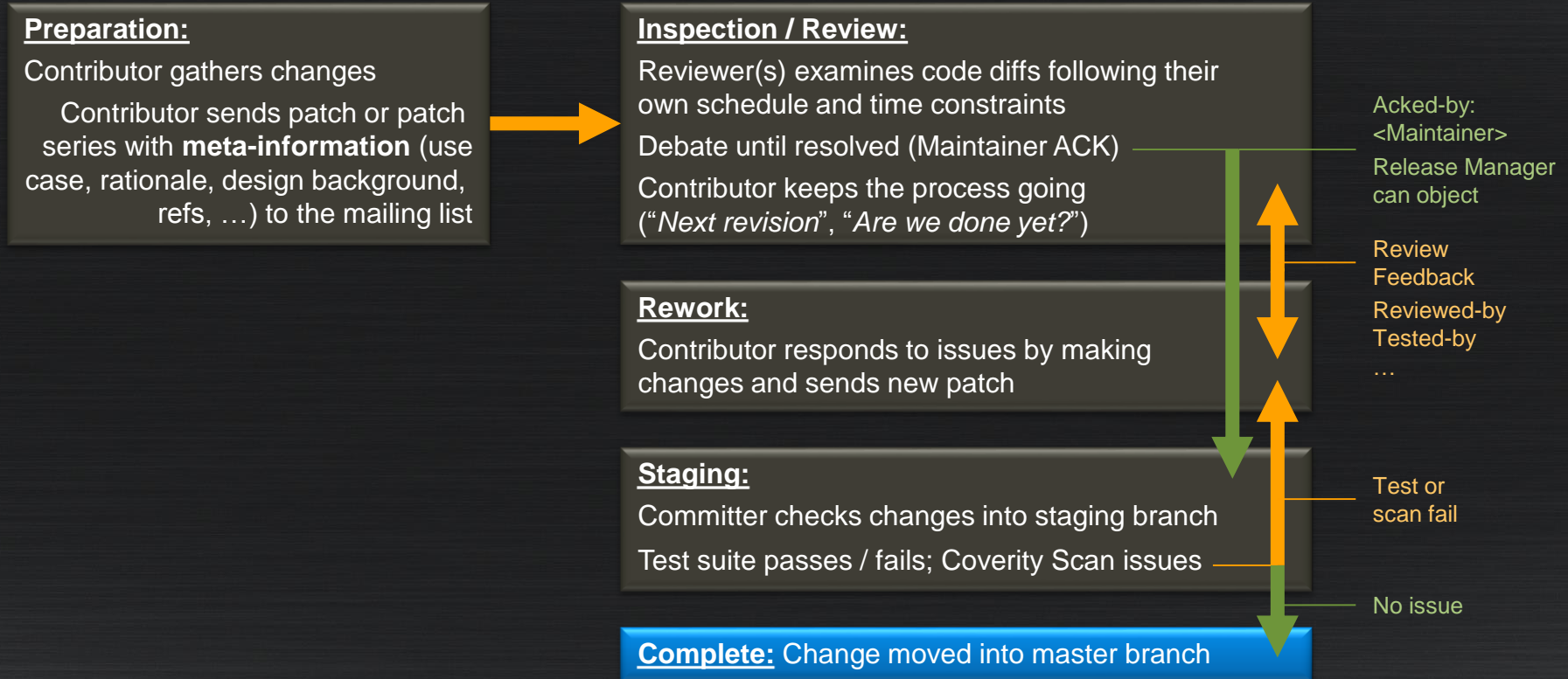


Contributions:

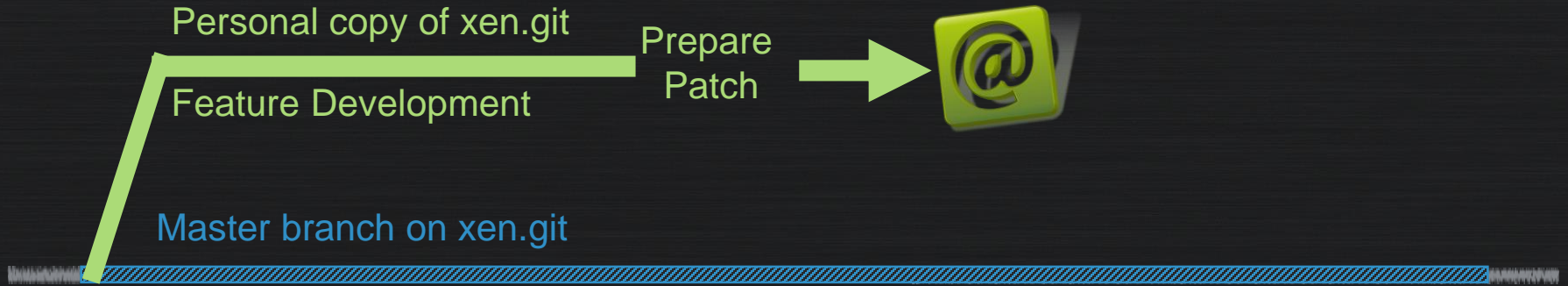
wiki.xenproject.org/wiki/Submitting_Xen_Project_Patches

www.xenproject.org/help/contribution-guidelines.html

E-mail based review Process



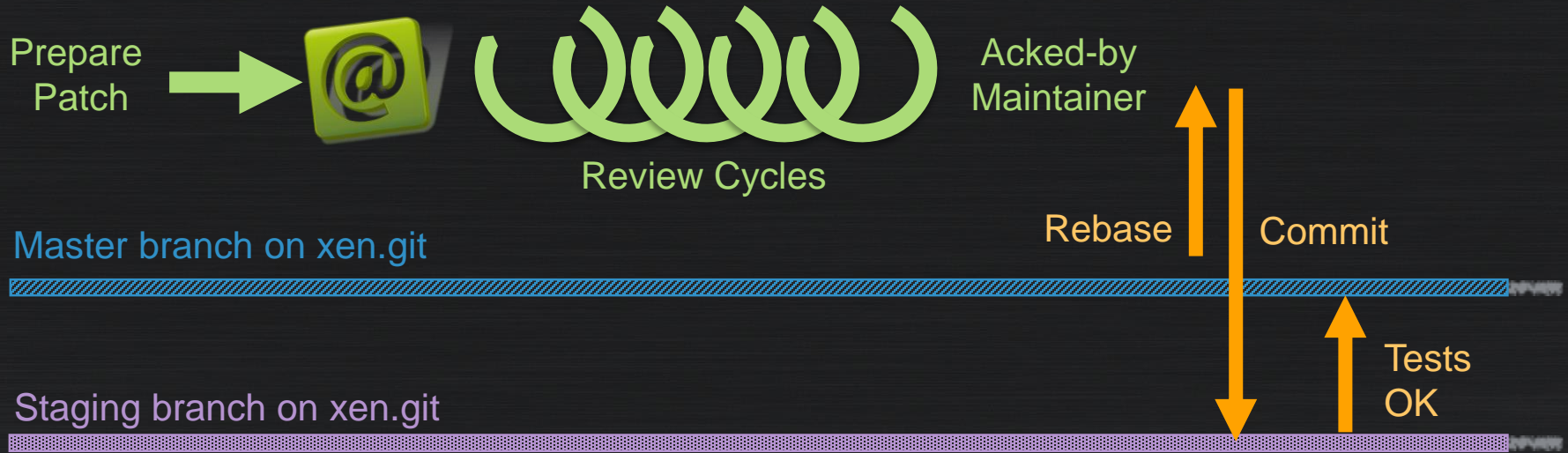
Review Process : Codeline View



Note: that you can request a personal development git tree to be hosted on xenbits.xenproject.org by sending an email to xen-devel@

These are listed on <http://xenbits.xen.org/gitweb/> under `people/*`

Review Process : Codeline View



Contains a limited number of “pending” patches
Identifies patches that lead to test failures in OSSTest via bisection.
Pushes patches that do not lead to regressions automatically to master



Anatomy of a Good Patch Series:

Or what makes a good patch (series)

Key parts of a Patch / Patch Series

- Description of patch series : provides
 - Information about its [use, use-case and users](#) (if not obvious)
 - Information about the [design, architecture, assumptions](#), etc. (if not obvious)
 - References to [relevant context information](#) - e.g. specs, etc. – (if relevant)
 - Short description of [elements of the patch series](#) and how they relate to each other (if necessary)
- Each patch contains
 - [Description](#) of the change and [what it is for](#)
 - Ideally also contains an analysis of the [problem solved and why it is solve this way](#)
 - The [patch](#) itself (ideally less than 200 LOC, reviewable in < 1 hour)
 - API documentation and test cases (if appropriate)
 - [Sign-off-by](#) : you state that you abide by the [Developer Certificate of Origin](#)

Issue: Unclear Description

marc.info/?l=xen-devel&m=141407695210809

This driver uses hwdom to change frequencies on CPUs

marc.info/?l=xen-devel&m=141407702410864

Xen changes frequencies on CPUs using this high-level cpufreq driver.

marc.info/?l=xen-devel&m=141492507021019&w=2

This patch series are only the Qemu part to enable Xen stubdom vTPM for HVM virtual machine. it will work w/ Xen patch series and seaBios patch series.

Build it with --enable-tpm and --enable-xen options and link with Xen, or change QEMU_STUBDOM_VTPM compile option from 'n' to 'y' in <Xen>/Config.mk, when the Qemu/ SeaBios patch series are merged.

Run Xen virtual machine with below QEMU command line options: "-tpmdev xenstubdoms,id=xenvtpm0 - device tpm-tis,tpmdev=xenvtpm0" or Xen xl tool adds this options when virtual machine cfg includes: **vtpm=["backend=vtpmN"]**

qemu-system-* --tpmdev help

Supported TPM types (choose only one): passthrough - Passthrough TPM backend driver xenstubdoms; Xenstubdoms -TPM backend driver

Example of a better description

lists.xenproject.org/archives/html/xen-devel/2014-10/msg00993.html

This patch series breaks multiboot (v1) protocol dependency and adds multiboot2 support. It lays down the foundation for EFI + GRUB2 + Xen development. [Detailed description of ideas and thoughts you will find in commit message for every patch.](#) If something is not obvious please drop me a line.

Patch #13 reveals a bug which probably was introduced between commit 3e2331d271cc0882e4013c8f20398c46c35f90a1 (VT-d: suppress UR signaling for further desktop chipsets) and 61fd8a7acf3de11f3d50d50e5b4f4ecfac7e0d04 (x86/HVM: properly bound x2APIC MSR range). Xen crashes at video_endboot() because earlier scrub process wipes vga_console_info data (sic!). So, it means that at least page containing this structure was freed mistakenly somewhere. Interestingly this issue appears on legacy BIOS machines only. EFI platforms work as usual. It is possible to workaround this bug by passing no-bootscrub to xen.gz.

I was not able to spot anything obvious just looking briefly at commit history. I am going to narrow down and fix this issue in next release.

ARM build has not been tested yet. Most of the requested things are fixed but there are still some minor outstanding issues (multiboot2 tags generation, excessive amount of casts in xen/arch/x86/boot/reloc.c, etc.; please check commit messages for more details). If something is not fixed yet it means that I do not have good idea how to do that. In case you spot something which was mentioned during previous reviews and still think that your comment is valid in particular case please notify me.

Example of a good description of an Individual Patch

lists.xenproject.org/archives/html/xen-devel/2014-11/msg00525.html

When using pvgrub in graphical mode with vnc, the grub timeout doesn't work: the countdown doesn't even start. With a serial terminal the problem doesn't occur and the countdown works as expected.

*Problem
Description*

It turns out that the problem is that when using a graphical terminal, checkkey () returns 0 instead of -1 when there is no activity on the mouse or keyboard. As a consequence grub thinks that the user typed something and interrupts the count down.

*Analysis
of why it occurs*

To fix the issue simply ignore keystrokes returning 0, that is the NUL character anyway. Add a patch to grub.patches to do that.

*How the patch
fixes it*

Example of a good Patch Series description

lists.xenproject.org/archives/html/xen-devel/2014-08/msg02369.html

The x86 architecture offers via the PAT (Page Attribute Table) a way to specify different caching modes in page table entries. The PAT MSR contains 8 entries each specifying one of 6 possible cache modes. A pte references one of those entries via 3 bits: `_PAGE_PAT`, `_PAGE_PWT` and `_PAGE_PCD`.

The Linux kernel currently supports only 4 different cache modes. The PAT MSR is set up in a way that the setting of `_PAGE_PAT` in a pte doesn't matter: the top 4 entries in the PAT MSR are the same as the 4 lower entries.

This results in the kernel not supporting e.g. write-through mode. Especially this cache mode would speed up drivers of video cards which now have to use uncached accesses.

OTOH some old processors (Pentium) don't support PAT correctly and the Xen hypervisor has been using a different PAT MSR configuration for some time now and can't change that as this setting is part of the ABI.

This patch set abstracts the cache mode from the pte and introduces tables to translate between cache mode and pte bits (the default cache mode "write back" is hard-wired to PAT entry 0). The tables are statically initialized with values being compatible to old processors and current usage. As soon as the PAT MSR is changed (or - in case of Xen - is read at boot time) the tables are changed accordingly. Requests of mappings with special cache modes are always possible now, in case they are not supported there will be a fallback to a compatible but slower mode.

Summing it up, this patch set adds the following features: - capability to support WT and WP cache modes on processors with full PAT support - processors with no or uncorrect PAT support are still working as today, even if WT or WP cache mode are selected by drivers for some pages - reduction of Xen special handling regarding cache mode

Patch series with Design Information

marc.info/?l=xen-devel&m=123003693614292

This set of patches introduces a set of mechanisms and interfaces to implement populate-on-demand memory. The purpose of populate-on-demand memory is to allow non-paravirtualized guests (such as Windows or Linux HVM) boot in a ballooned state.

BACKGROUND

When non-PV domains boots, they typically read the e820 maps to determine how much memory they have, and then assume that much memory thereafter. Memory requirements can be reduced using a balloon driver, but it cannot be increased past this initial value. Currently, this means that a non-PV domain must be booted with the maximum amount of memory you want that VM every to be able to use.

Populate-on-demand allows us to "boot ballooned", in the following manner:

- Mark the entire range of memory (`memory_static_max` aka `maxmem`) with a new p2m type, `populate_on_demand`, reporting `memory_static_max` in the e820 map. No memory is allocated at this stage.
- Allocate the "`memory_dynamic_max`" (aka "target") amount of memory for a "PoD cache". This memory is kept on a separate list in the domain struct.
- Boot the guest.
- Populate the p2m table on-demand as it's accessed with pages from the PoD cache.
- When the balloon driver loads, it inflates the balloon size to (`maxmem - target`), giving the memory back to Xen. When this is accomplished, the "populate-on-demand" portion of boot is effectively finished.

...



Anatomy of a Good Patch Revision:

Addressing Feedback!
Common mistakes to avoid!

Patch Revisions

Each new patch revision contains

- **Change history** (what has changed between each iteration)
- Tags (by maintainers, reviewers, testers, etc.)
 - Acked-by: <Maintainer>
 - Reviewed-by: <Reviewer>
 - Tested-by: <Tester>
 - Note that there is disagreement about use of Acked-by: <non-maintainer> which is being discussed amongst the community and not yet resolved (see <http://lists.xen.org/archives/html/xen-devel/2014-06/msg01419.html>)

Issue: Changes that were not asked for

marc.info/?l=xen-devel&m=140259327023935

```
> Add support for GIC v3 specification System register access (SRE)
> is enabled to access cpu and virtual interface registers based
> on kernel GICv3 driver.
>
> This patch adds only basic v3 support.
> Does not support Interrupt Translation support (ITS)
```

I see that a lot of things changed in this version of the patch. It would be nice if you kept a log of the incremental changes.

In particular in this version of the patch we basically added `isb()` everywhere. I would like to see a written comment about it.

When Julien commented "isb?", I think he was just asking whether they are needed, not requesting you to add them. The only one I was sure about was the one at the end of `gicv3_eoi_irq`.

But it is good to see that you addressed all my comments.

Makes it **hard** for the reviewer to only re-review the portions of the patch that need to be looked at

Seems there was also a misunderstanding between reviewer and submitter


Issue: Incomplete Resend

marc.info/?l=xen-devel&m=140285195123082

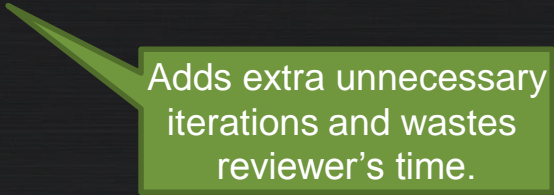
You didn't address the comments I made on V4 for this patch. See a copy of them inline...

marc.info/?l=xen-devel&m=140422021120851

You continue to ignore my comments on this patch... Please explain why you don't want to make those changes. Sending 3 times the same patch without any change is a waste of time for both of us.



When you don't understand
understand feedback,
feel free to ask for clarification



Adds extra unnecessary
iterations and wastes
reviewer's time.

Example of a good changelog

v3 - suggestions/fixes:

- rename some variables
(suggested by Andrew Cooper),
- remove unneeded initialization
(suggested by Andrew Cooper),
- improve comments
(suggested by Andrew Cooper),
- further patch split rearrangement
(suggested by Andrew Cooper and Jan Beulich).

v2 - suggestions/fixes:

- improve inline assembly
(suggested by Andrew Cooper and Jan Beulich),
- use `__used` attribute
(suggested by Andrew Cooper),
- patch split rearrangement
(suggested by Andrew Cooper and Jan Beulich).



Coding Styles:

Hypervisor Coding Style

xenbits.xen.org/gitweb/?p=xen.git;a=blob;f=CODING_STYLE

- Indentation
- White Spaces
- Line Length
- Bracing
- Comments
- Emacs Local Variables

Libxl Coding Style

xenbits.xen.org/gitweb/?p=xen.git;a=blob;f=tools/libxl/CODING_STYLE

Similar to QEMU and Linux with a few exceptions. Different from Hypervisor coding style.

- Whitespaces
- Line width
- Variable Naming
- Statements
- Block Structures

Tools to check coding styles

There are a number of uncommitted scripts, which automatically check coding style. These [have not yet been fully reviewed and may not work in all cases](#), but may partly be usable. See

- lists.xen.org/archives/html/xen-devel/2014-09/msg01543.html
- lists.xen.org/archives/html/xen-devel/2014-09/msg01171.html

Other useful scripts:

- [scripts/get_maintainer.pl](#) : looks at the files you have modified in the patch and matches it up with the information in the MAINTAINERS file → [generates a list of people + email addresses who need to be CC'ed](#)
→ Some issues: see lists.xenproject.org/archives/html/xen-devel/2014-11/msg00060.html



Release Process:

blog.xenproject.org/2014/05/08/how-the-hypervisor-team-manages-releases/

Roles: Release Manager

- A Maintainer that is **elected** by the community to manage a specific release
 - Nominated by community members (but needs to have agreed to want to do the job and have the time to do so)
 - Elected using a **formal vote** by committers of the Xen Project Hypervisor team
 - For one or several releases
 - (Past) Release Managers: Ian Campbell (Citrix), George Dunlap (Citrix)
 - Xen 4.5 Release Manager: **Konrad R Wilk (Oracle)**
- Responsible for
 - Driving the **Roadmap process**
 - Deciding the **Release Timetable**
 - Making the Final call on **what goes into the Release** at each Freeze Point
 - Handling requests for “freeze” **Exceptions** : Cost vs. Benefit analysis
 - **Coordinates** with other committers in making Release Candidates and Releases
- Other responsibilities
 - **Helps with PR** related to the release
 - Can also act as **spokesperson** for the project (if the Release Manager is allowed to by employer)

Release Process Goals

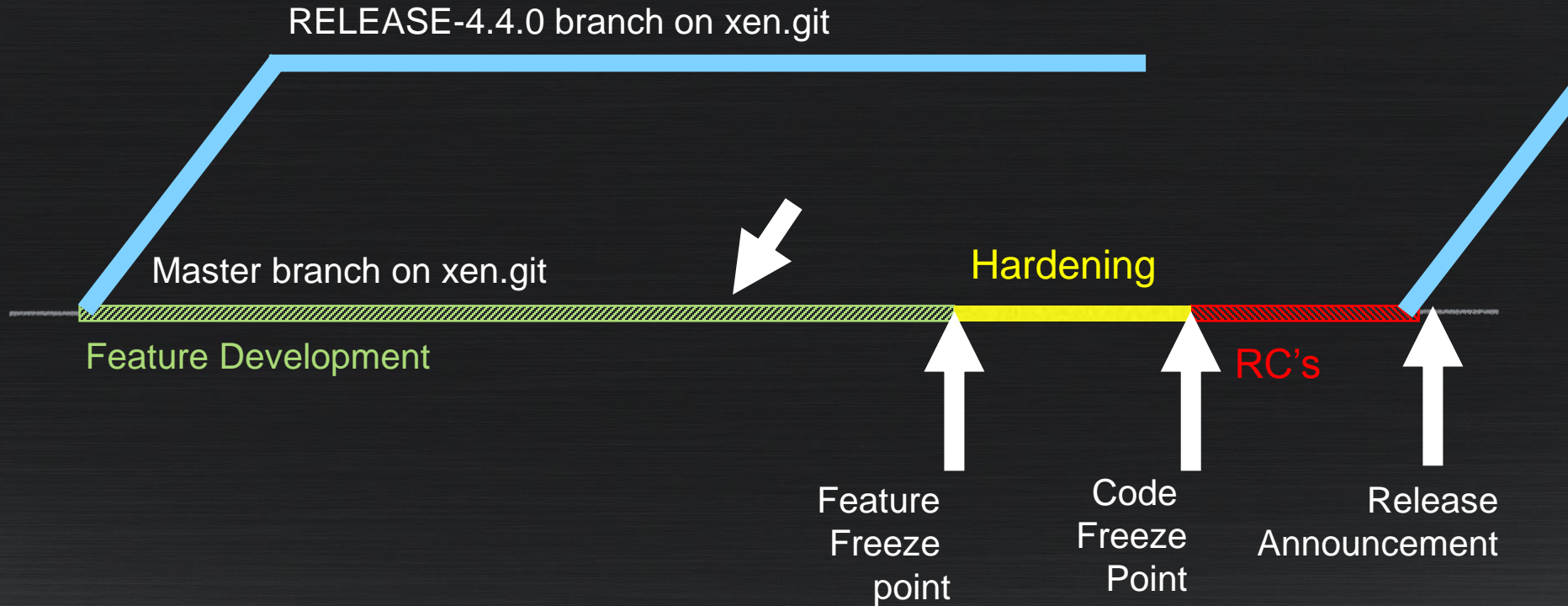
A bug-free release

An awesome release

An on-time release

The Xen Project aims to do releases on a **6 month cadence**, but sometimes we go for exceptions to allow for bigger features to go in (**e.g. Xen 4.5 is 9 months**)

Release Process : Stages



Stages & Gates in more detail

Development

- This is when **patches** for the ongoing release **need to be submitted for review**

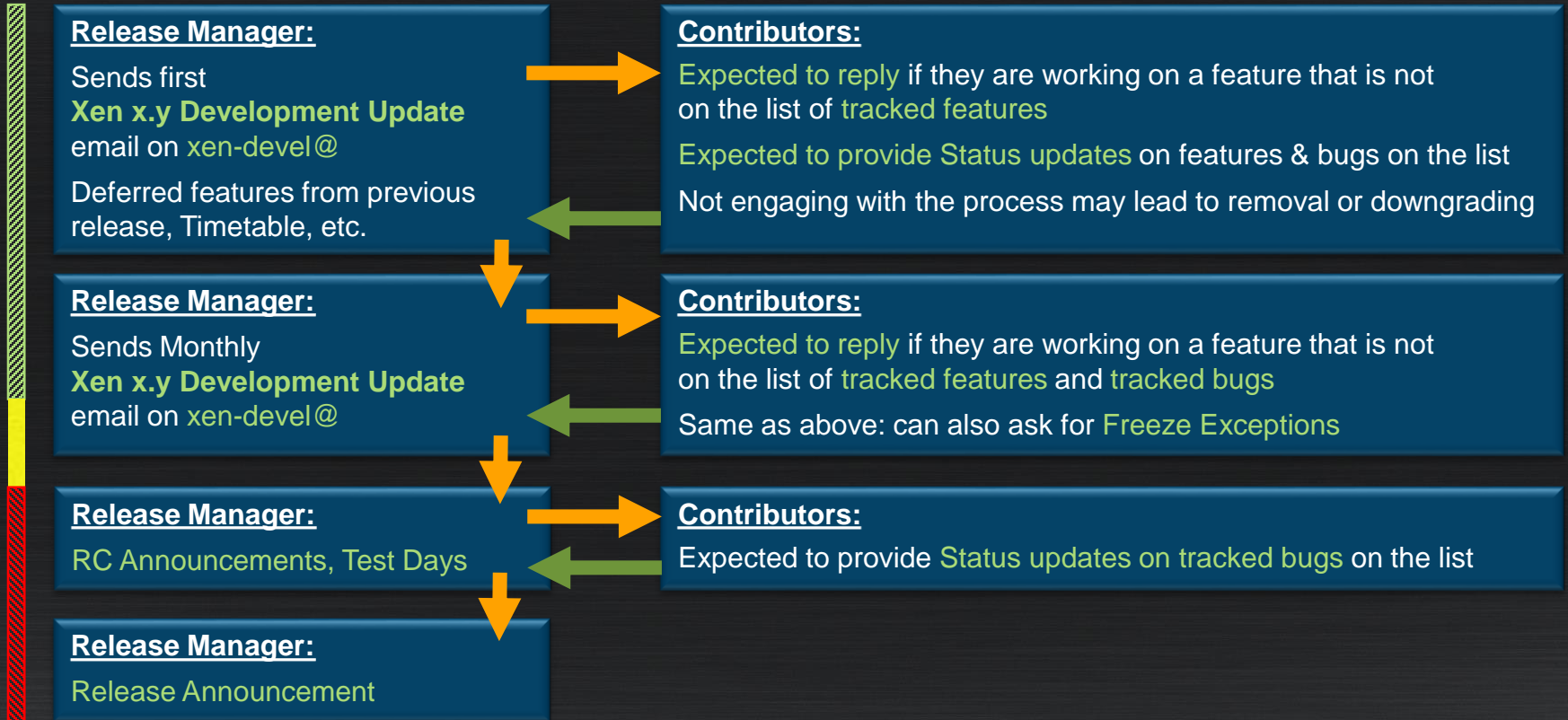
After Feature Freeze / Hardening:

- No **new** Features
- The focus of Hardening is on **Bug Fixes**
- Patches that were **submitted before Feature Freeze** and are still being reviewed **can still be let in** based on a risk analysis (see exceptions)

After Code Freeze / RCs:

- **No bug fixes** will be accepted, **unless** they are deemed a **blocker**

Release Process : Tracking Contributions



Criteria and Considerations

The release manager's decision on a patch going in or out / after feature freeze date is in huge part influenced by:

- Risk of regressions:
 - Does this patch introduce a regression that will **affect a large set of use-cases**
 - Does this patch touch **common code areas** (code used by more components is less likely to be accepted, as it is more risky)?
 - Does this code affect **specific cases only** (e.g. MMIO pass through only) and is thus less risky?
 - Is this code **simple and easy to understand** and thus less risky?
- Has the code **been tested for failures and succeeded multiple times?**
- Has the code been **Acked or Reviewed** by the maintainer?

Release Process : Freeze Exceptions

When approaching Feature Freeze and Code Freeze points, **features and bug fixes are deferred into the next release**

Contributors can request Freeze Exceptions by replying to the **Xen x.y Development Update** mail. Freeze exceptions are part of the **normal release process** and trigger a **cost-benefit analysis** by the Release Manager with input from the community.

A Feature Freeze Exception

- Patch must already have been **posted for review**
- An **explanation** must be given as to why an exception should be granted and why you believe the risk to the release quality will be acceptable
- Typically maintainers and the Release Manager will assess
 - **Likelihood of the code being completed** before the next milestone
 - The **risk** the code introducing quality issues, security issues, performance regressions, etc.
 - You may be asked to **take risk mitigating measures**: such as provide extra Test Code, etc. to prove low risk

A Bug Freeze Exception

- Similar as with a Feature Freeze exception

Also see: blog.xenproject.org/2014/06/19/release-management-risk-intuition-and-freeze-exceptions/

Release Process : Roadmap

Roadmap evolves as a consequence of **Xen x.y Development Update** emails on xen-devel

Wiki page: wiki.xenproject.org/wiki/Xen_Roadmap/x.y

- links to mails and relevant resources
- maintained on best effort basis

Reading the “Roadmap”:

The "prognosis" states the likelihood of completion of features in the **x.y timeframe**

None: Nothing yet

Fair: Still working on it, patches are prototypes or RFC

Ok: Patches posted, acting on review

Good: Some last minute pieces

Done: All done, might have bugs

} *Possibility that a Feature Freeze exception may be granted*

A feature marked as “Done”, will go into the currently developed release unless it turns out that it introduces significant quality issues that cannot be resolved in the current release cycle



Testing & Coverity Scan:

Coverity Scanning Service

The Xen Project is registered with the "Coverity Scan" service which applies Coverity's static analyser to the Open Source projects

Because "Coverity Scan" may discover security issues the full database of issues cannot simply be made public.

To get access, consult

- www.xenproject.org/help/contribution-guidelines.html under "Code Security Scanning"

OSSTEST

The Xen Project has an automated test infrastructure that is run on

- Xen.git master
- Xen.git staging (pushgate) – see earlier
- Overview:
 - www.xenproject.org/help/presentations-and-videos/video/xpds14-osstest.html
 - xenbits.xen.org/gitweb/?p=osstest.git;a=blob;f=README;hb=HEAD

Can be run in stand-alone mode

- blog.xenproject.org/2013/09/30/osstest-standalone-mode-step-by-step/



Earning Status:

Earning Maintainer & Committer Status

Maintainers

- Demonstrate **good technical knowledge** in an area and submitted good patch series **over a prolonged time period** (6-12 months)
- Demonstrates **engagement and engagement skills in the community via reviews**
- **Self-nominates**, then signed off by other Maintainers

Committers

- Demonstrates prolonged engagement with the community **upholding technical and community values** – in other words **demonstrates that he/she can be trusted**
- Needs to be one of the **top contributors** to the project **for a prolonged time period** (e.g. >200 patches submitted per year; no objective rules)
- Gets **elected** by other maintainers



Meetings:

Face-2-face meetings and other community activities

IRC: #xendevel

- Invite only to avoid user questions on #xendevel
- BUT: anyone who has submitted a few patches can get access
- Ask for an invite on xen-devel@
- A lot of **coordination, quick pings, reminders** happen on #xendevel

F-2-F: Xen Project Hackathons

- Once per year, hosted and paid for by a vendor
- 2 days long, 25-45 people depending on location
 - 95% developers
- Not so much a Hackathon, but a series of structured 1-1.5h discussions to solve **architectural**, **design**, **review**, **process** and **other issues**
 - Typically 3-6 discussions happen in parallel
 - Sometimes presentations are used, but **interaction**, **discussion** and **decision making** are the key focus of Hackathons
 - Discussions are minuted and posted on mailing lists
- Also see: blog.xenproject.org/2013/05/28/event-report-xen-hackathon-2013/

F-2-F: Xen Project Developer Summits

- Once per year, organized by the Xen Project
 - Co-located with LinuxCon Europe or North America
- Paid for by sponsorship and tickets, and subsidized by Xen Project
- 2 days long, 90-120 people depending on location
 - 70-80% developers
- Mainly presentations and conference format
 - Some interactive elements (BoF discussions)

F-2-F: Xen Project Developer Meeting

- Once per year, organized by the Xen Project
- Co-located with Xen Project Developer Summit (the day before or after)
- Typically we also have Working Group meetings and Advisory Board meetings on the same day
- 3 hours long, 20-25 core developers, maintainers and committers
 - Plenary format
 - 6-10 topics covered

Calls: Technical Coordination Team Meeting

- Monthly conference call
- Invites on request
- Opening up the format (currently being discussed) to make it more accessible

Calls: Ad-hoc Meeting

- Community members can propose ad-hoc conference calls (or other on-line meetings) on specific topics on development lists
- Community Manager will set up details as needed
- Can be a conference call, IRC meeting, etc.