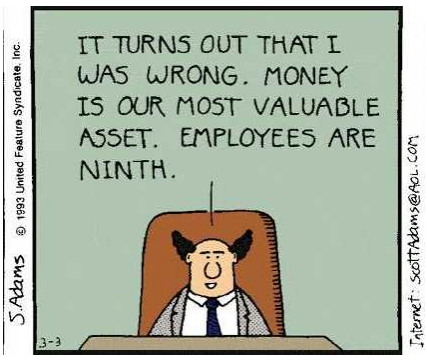
Corporate
Open Source



What motivates business decision makers: $$$

Is there a big enough market?

How do we get our customers to pay us?

What's the minimum viable project to get them to pay us?

How do I protect my "turf" from competitors?

Why do companies do open source?

- Competitive advantage
- Common shared resources
- Sell hardware, services, or data
- Customers require it

Why do companies do open source?
Competitive advantage
They sell hardware, services, or data and give away software
Their customers demand it

$$$ thinking leads to common pitfalls that violate key principals of the open source community

Open source devs see repeating patterns in corporations that try to participate in open source

Is there a way for developers (and product managers, evangalists, tech writers, etc) convince the business decision makers how to do open source right?

Yes and no, because it's very slow to change corporate culture and...

...sometimes people have to get burned a couple times.

Explain hazards and BKMs

Won't help until they get burned

Cultural change is hard

Unless things get painful, they will stick with what they know

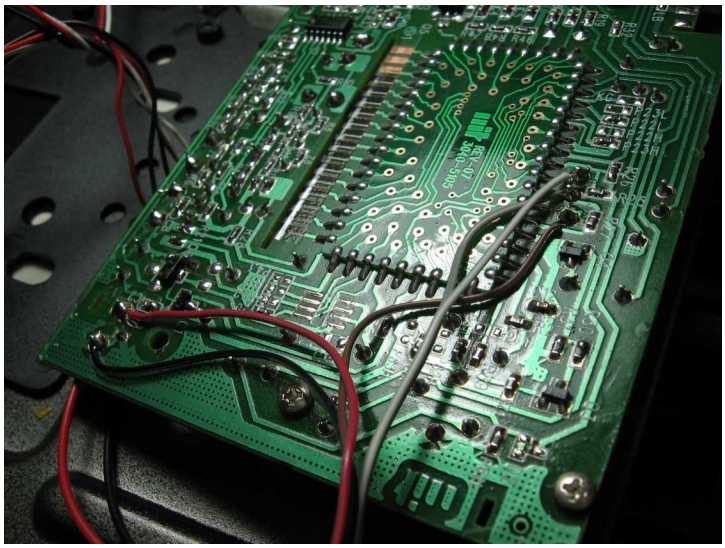
How do you motivate decision makers to change?

Open source means
transparency

Binary Blobs

- Closed source binaries in open source projects
- The locked box where you keep your secrets
- Licensing, FCC regulations, or IP concerns
- Fear of competition - "If we share this secret sauce, our competitors will win!"

Binary Blobs



Reverse engineering (tracing with debuggers or packet sniffers, disassembling, hijacking the firmware update process, poking at security holes)

good luck explaining this to managers

Binary Blobs

Socratic questioning -
  What exact IP are we hiding? Are there patents for that IP we need to protect? Would someone "versed in the art" be able to recreate our IP? What is the business impact if that IP was leaked? Are there already open source projects available that do most of the same things?
Work with lawyers to understand your legal requirements - break through the FUD of what "might happen"

Open source developers
release frequently

Upstreaming Technical Debt

“We'll just fork it"

Effort to upstream and rewrite or rearchitect is an unnecessary cost in biz dev's eyes

If you're always building a new product, it's easy to fork and forget.

Upstreaming is more cost-effective for businesses that build on the same code base for future projects, or re-use features in a fast-moving code base.

Upstreaming Technical Debt

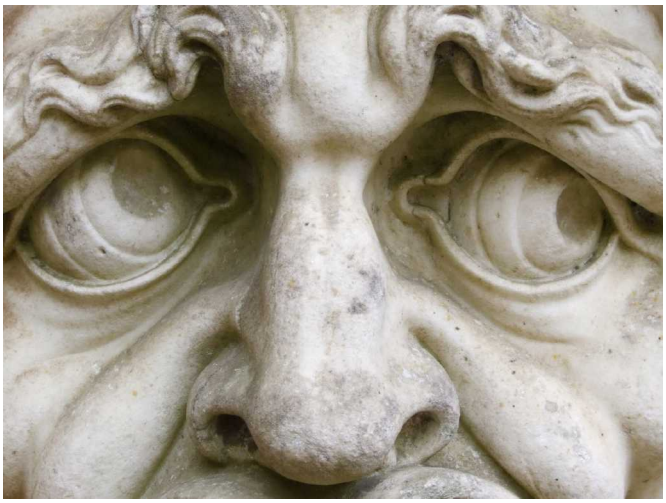
- Hard to update older designs, if you have technical debt.
- Technical debt builds up
- Customers don't get the latest software, impacting their security and time to market
- Unhappy customers go elsewhere, impacting your bottom line

Upstreaming Technical Debt

- Short term: convince your boss to eliminate the technical debt that's most difficult to maintain.
- Long term: build a relationship with upstream to make code submissions easier

Open source developers
collaborate

Creating perfection...



Many developers are worried about their code being public

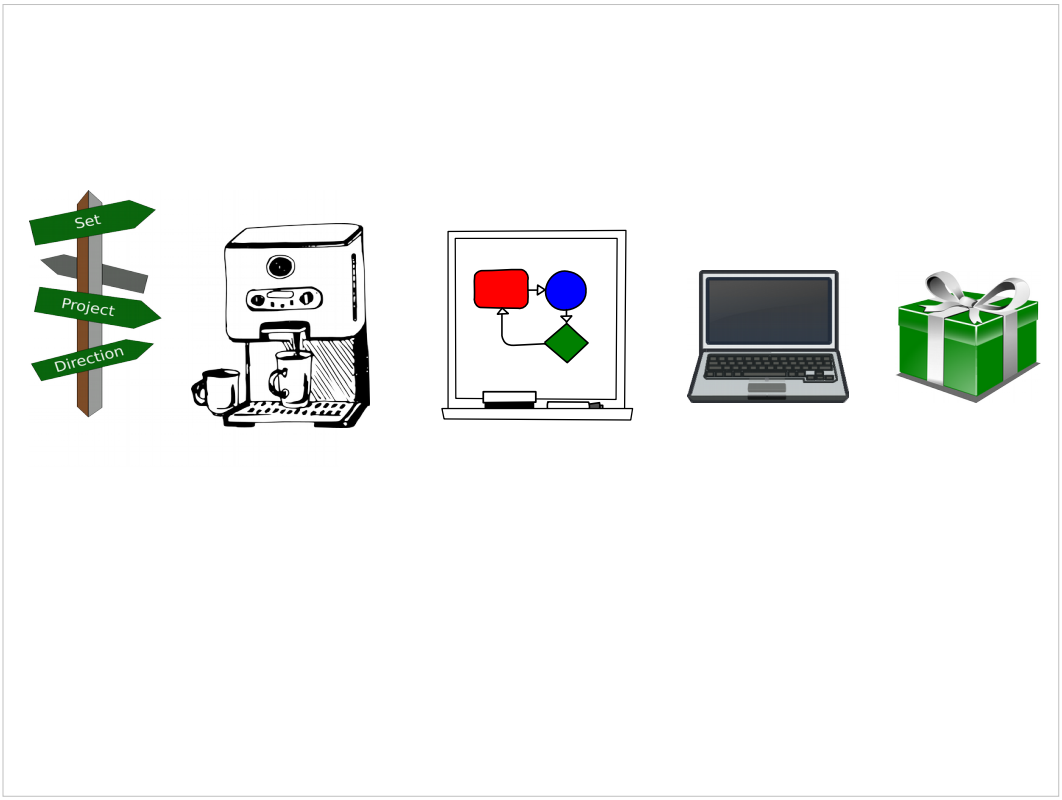
They spend a lot of time polishing their code and trying to get it to be “perfect”

Creating perfection... or polishing turds?

Many developers are worried about their code being public

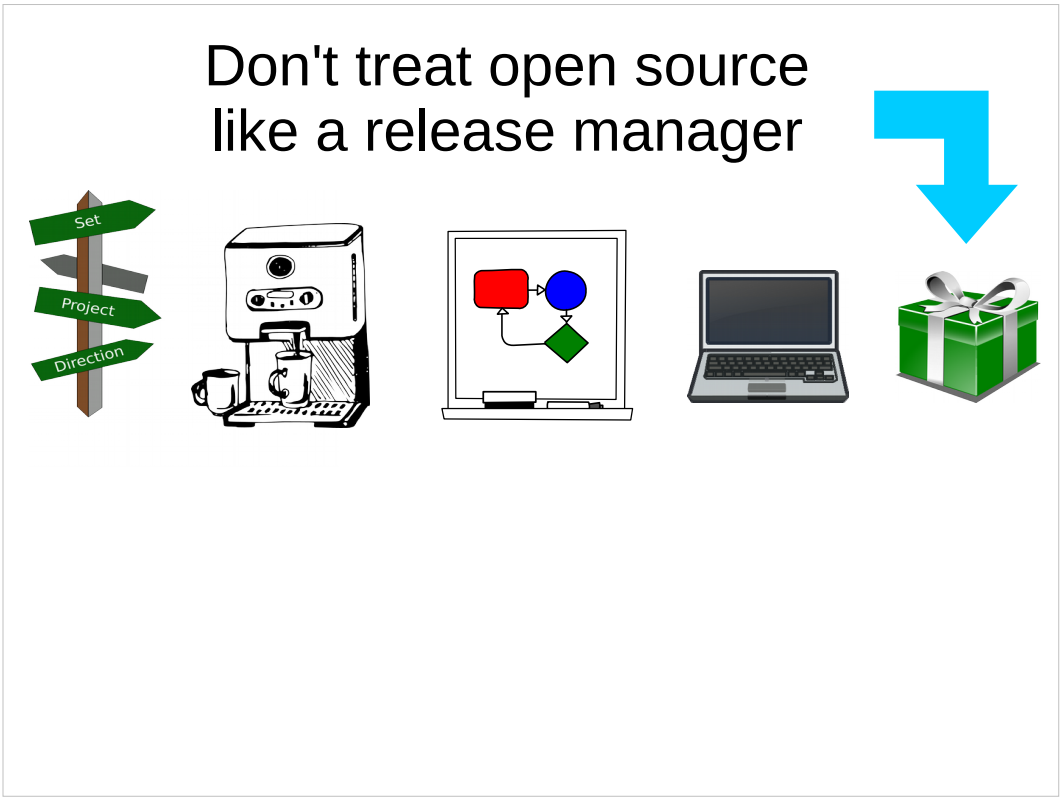
They spend a lot of time polishing their code and trying to get it to be “perfect”

Open source developers
define project direction



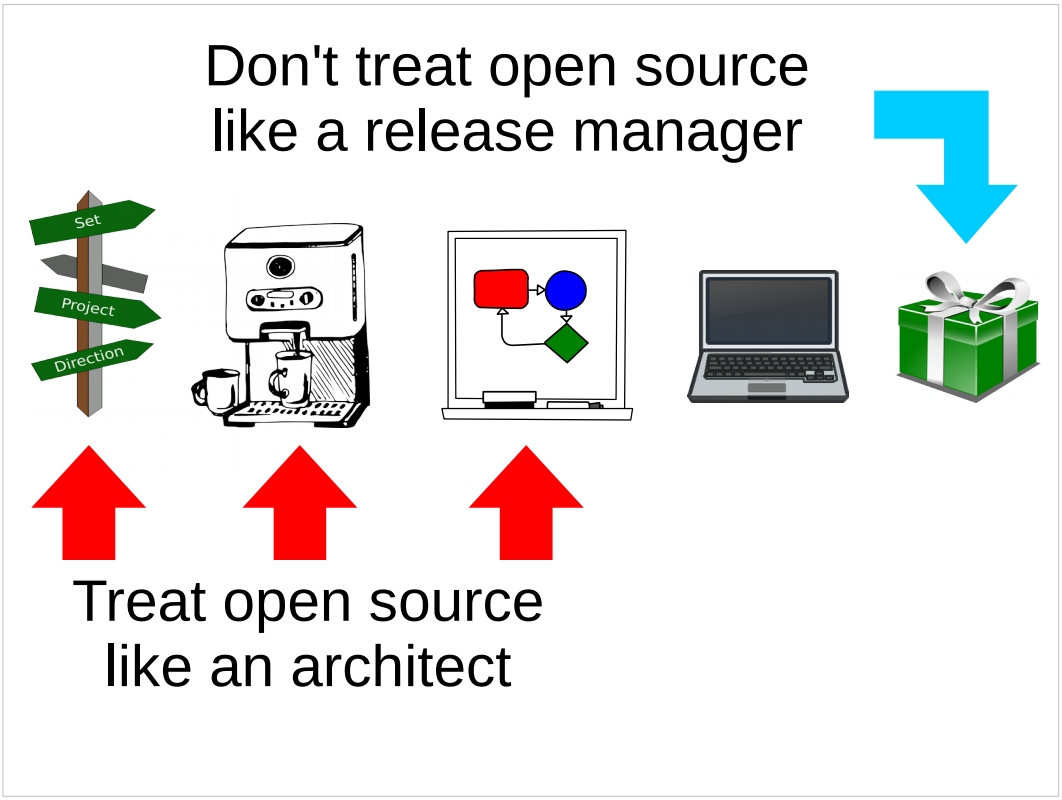
Short term: convince your boss to eliminate the technical debt that's most difficult to maintain.

Long term: build a relationship with upstream and get your code in early, not during product release crunch time.



Short term: convince your boss to eliminate the technical debt that's most difficult to maintain.

Long term: build a relationship with upstream and get your code in early, not during product release crunch time.



Short term: convince your boss to eliminate the technical debt that's most difficult to maintain.

Long term: build a relationship with upstream and get your code in early, not during product release crunch time.

Open source projects
have release schedules

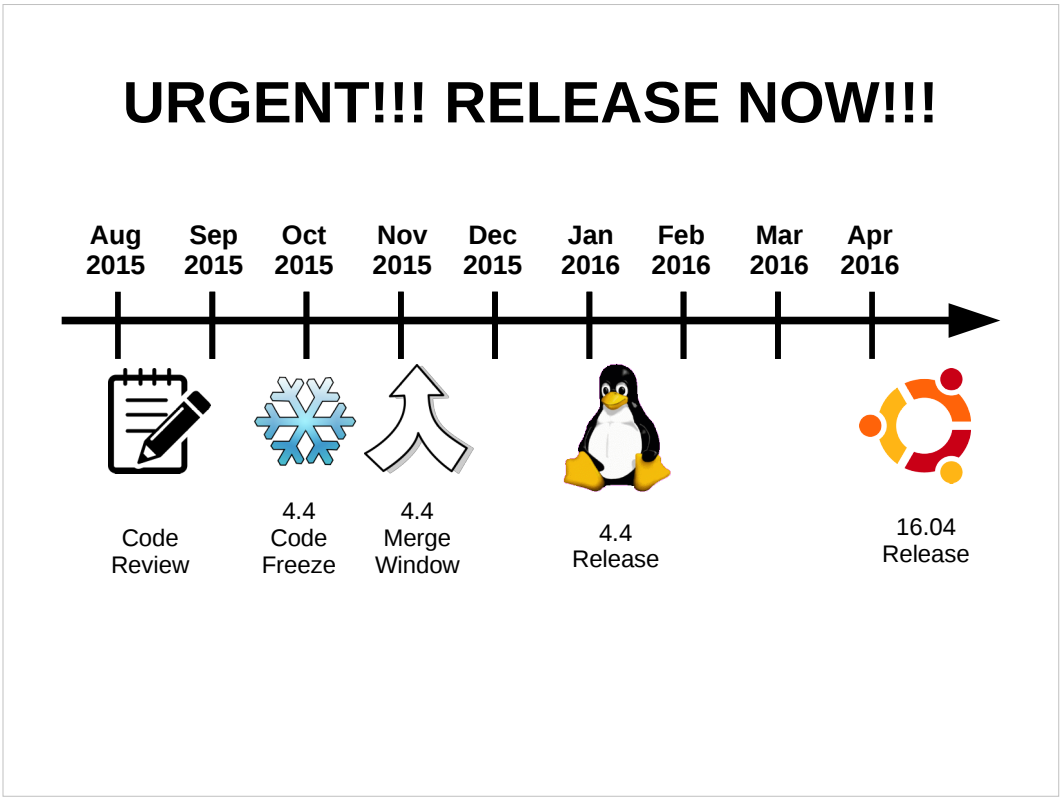
URGENT!!! RELEASE NOW!!!

Management and project managers are often pushing engineers to release

In English, we have a saying, “Your fire is not my emergency.”

URGENT!!! RELEASE NOW!!!

- Your corporate timeline is not something the open source community cares about
- Project managers often treat open source as release managers, not partners
- They imagine that once the code is merged or even made public, everyone suddenly has it. Takes time to get into distros.



Need to educate management about FOSS code freeze & release deadlines

Example: get a webcam kernel driver in Ubuntu 16.04 (April 2016 release)

Kernel 16.04 shipped with was 4.4 based v4.4.0 shipped in Jan 2016

4.4 code merge window start Nov 1 2015

code approved 4.3-rc4 - Oct 4 2015

Need 3-8 weeks code review & re-architecting - August or Sept 2015

9 months from start of code review to in a Linux distribution

Open source projects
accept contributions

Available Source != Open Source

Open source is more than a license.
It's about collaboration.
Some customer wants it to be "open source" to use, but doesn't care about giving back
Throw it over the wall:
Open in name only, no external contributions accepted.
One or two companies participating in a project
"Minimum viable product" to satisfy the "open source requirement"

Available Source != Open Source

Failure to bring outside perspective
Missed market opportunity from narrow thinking or insular requirements

Available Source != Open Source

What can we do to change this?

Internal pressure to collaborate more often doesn't work

Need external motivation:

Product flops because it doesn't address all customer needs

Other companies start a competing group or project

Bad press and constant pressure from external communities crucial to biz needs

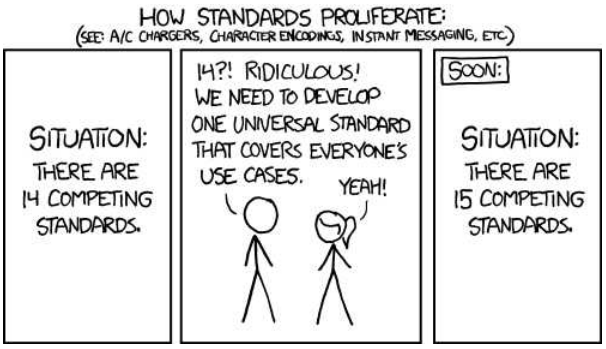
Open source encourages
the best technical solutions

Joke: of course, everyone has different opinions about what the “best” solution is

Reinventing the Wheel

- Everyone is trying to invent the next big thing
- Customers will try out the latest shiny new watch just for fun
- Companies often assume open source is like that
- VPs and marketers think developers will try out a new open source project, fall in love, and immediately switch to the latest web framework

Reinventing the Wheel



http://xkcd.com/927/

And that means developers have another new project to evaluate

Reinventing the Wheel

- Disrupting an existing open source community is hard.
- Developers are quick to experiment with a new open source project, but uptake in shipping products is very slow.
- You need to have a track record of support, consistent updates, and the features your customers really need.

No amount of marketing $$$
will buy developer trust
in your open source community

It takes years of hard work (writing articles and blog posts, answering questions on hackernews and IRC, and a constant community presence) in order to make an open source project successful.

No marketing $$$ can duplicate that

Instead, companies are buying core developers to promote their products.

Good? Bad? Discuss

Avoiding Corporate Open Source Fail

- Question hiding info
- Reduce technical debt
- Treat open source as architects
- Submit early, submit often
- Disruption takes time & community

Thanks!

Sarah Sharp

Otter Tech
Open Source Training
sharp@otter.technology