A Chance Gardener

HARVESTING OPEN-SOURCE PRODUCTS AND PLANTING THE NEXT CROP

Dear KV,

I'm working at a startup where we use a lot of open-source code software, not just for our operating systems, but also at the core of several products. We've been building our systems on top of open source for several years, but at this point we only consume the software, we never have time to contribute patches. Working with 30 to 40 different projects, given our small staff, would introduce a lot of engineering overhead that the company simply cannot absorb at the moment.

It also seems to me and the rest of what passes for management at our company that open source is like a massive garden of weeds. New projects pop up all the time, and it's impossible to know if these are really harmful or helpful to our overall systems, so we have to try them out or risk being left out of some new type of system. The other day one of the engineers complained that he felt like a gardener whose only tool was a machete, which is not a precision tool.

Larger companies clearly know how to work with open-source projects, but how can a startup or even a medium-size company, which lacks the resources to look at all this stuff, cope in the open-source world. What's the best way to interact with all these projects?

A Chance Gardener

who is
KV?

click for video

Dear Chance,

You have hit upon an excellent metaphor for open-source software: a garden. I have to admit, I might liken it more to *kindergarten*, but let's proceed with your original metaphor.

Many people who have not worked directly with open source assume that it is a single thing, or a single idea, when, in fact, it is a term that is applied in as many different ways as there are open-source projects and communities. Open source truly is like a garden, one with many different species of plants, some of which are beneficial and give nourishment and others of which are poison.

Separating the wheat from the chaff in such a large and diverse ecosystem is a nontrivial undertaking, but it is one that KV has addressed in several previous columns, including the letter to “Acquisitive”¹ (May 2015). Deciding to use a piece of software always comes down to the quality of the software in question, whether the software is closed, open, or somewhere in between. I find your question is more intriguing from the standpoint of how one interacts with open-source projects.

You mentioned that your company consumes open source, and, in fact, this is what most people and companies do—consume—and this is the first stage of working with open source. When you are consuming open source, the most important thing to remember is not to sever the plant from the roots. You should be consuming the software directly from the source, even if you are not following every single change to the upstream source tree.

The worst thing you can do is copy the source tree once

Mixing your changes directly into the tree, instead of on a development branch, is a great way to make any update to the software nearly impossible.

and then ignore upstream development for a period of time. Letting your local tree get even a few months out of date on a fast-moving project means you are missing a large number of changes and bug fixes. Often the bug fixes are also security fixes, and we all know what happens when people build products without proper integration of security fixes. Nothing. That's right, pretty much everyone gets a pass because we all know software breaks, and there is currently no liability for building insecure products.

Another way to sever the roots between your system and the open-source projects from which you consume code is to make your own changes in the master branch of the tree. Mixing your changes directly into the tree, instead of on a development branch, is a great way to make any update to the software nearly impossible. A great way to make sure you have not severed your software from the root of the tree is to have your own internal CI (continuous integration) system. Many open-source projects have their own CI systems, which you can directly integrate into your own development systems, and they can verify whether you have broken the system or if the upstream software itself is broken.

Continuing to stretch the metaphor, perhaps close to the point of breaking, we can think now about the next stage of tending the open-source garden. If you had a vegetable garden, but you never tended it, you would get few, if any, vegetables from the garden and it would wither and die. Open-source projects are no different in this respect from vegetables: we must tend the garden if we expect it to remain productive; otherwise, we're just being destructive.

There are many ways to tend a garden. Perhaps the first thing that comes to mind is weeding, which we might think of as debugging and patching. Contributing patches back to an open-source project is a way to help it improve and grow strong. Most open-source projects have a defined process whereby code contributions can be made. Although you mention the overhead involved in having your developers contribute to a project, you should turn this thinking on its head and realize that what they're doing when they submit patches to the upstream project is reducing your company's technical debt. If you keep a patch private, then it must be reintegrated every time you consume a new version of the open-source code. After a while, these patches can number in the tens of thousands, or more, lines of code, which is a huge amount of technical debt for you to maintain.

After some period of having your developers email patches and submit pull requests, you will realize that what you want on some projects are your own gardeners. Having members of your team working directly on the open-source projects that are most important to the company is a great way to make sure that your company has a front-row seat in how this software is developed.

It is actually a very natural progression for a company to go from being a pure consumer of open source to interacting with the project via patch submission and then becoming a direct contributor. No one would expect a company to be a direct contributor to all the open-source projects it consumes, as most companies consume far more software than they would ever produce, which is the bounty of the open-source garden. It ought to be the goal

Related articles

Forced Exception-Handling

You can never discount the human element in programming.

Kode Vicious

<https://queue.acm.org/detail.cfm?id=3064643>

Outsourcing Responsibility

What do you do when your debugger fails you?

Kode Vicious

<https://queue.acm.org/detail.cfm?id=2639483>

Using Free and Open-source Tools to Manage Software Quality

An agile process implementation

Phelim Dowling and Kevin McGrath

<https://queue.acm.org/detail.cfm?id=2767182>

of every company consuming open source to contribute something back, however, so that its garden continues to bear fruit, instead of rotting vegetables.

KV

References

1. Neville-Neil, G. Lazarus code. ACM Queue. 13[5]; <https://queue.acm.org/detail.cfm?id=2773214>.

Kode Vicious, known to mere mortals as George V. Neville-Neil, works on networking and operating-system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating

systems, and rewriting your bad code [OK, maybe not that last one]. He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. Neville-Neil is the co-author with Marshall Kirk McKusick and Robert N. M. Watson of *The Design and Implementation of the FreeBSD Operating System* [second edition]. He is an avid bicyclist and traveler who currently lives in New York City.

Copyright © 2018 held by owner/author. Publication rights licensed to ACM.