Scaling Your Developer Community via Plugins

Greg Sutcliffe
Foreman Community Lead
Red Hat

Mastodon: @gwmngilfen@fosstodon.org
You all know this already, right?
Kohsuke Kawaguchi (KK)  
Creator, Jenkins  

➢ “How we made the Jenkins community”  
➢ FOSDEM 2013 keynote  

https://www.slideshare.net/kohsuke/building-developer-community
“If you’ve said something 1,000 times, someone is still hearing it for the first time”
But what do I know?

- Contributor to TheForeman for 7 years
- Core developer for 4 years
- Community manager for 3 years
- Help out in numerous other communities

But mainly...
- I write a lot of plugins 😊
What do we want?

• More contributors!
  – Project health
  – Diversity in decision making

• But developers don’t come from nowhere
  – Employment
  – Bug Bounties

• Takes money
  – Smaller projects can’t do that
User Developer Funnel

Stranger

User

Developer

Core

“You don’t see the lost opportunities” - KK
What do we want?

● So we just need more users?
  - Marketing
  - Conference presence
  - Meetups

● Money again
  - Still not going to work
So what can we do?

• Focus on the funnel
  - Make it “wider”
    • i.e. easier for people to move down

• Harness desire
  - Desire to customize
  - Desire to contribute back
So what **can** we do?

**Foreman Community Survey 2018: Do you contribute to Foreman?**
Motivation Recap

- We want contributors
- Users want to contribute
- Growing either community directly takes money we don’t have

Focus on barriers to entry
Plugins are a good fit

• Assuming someone wants to code...
  • Help a new coder get used to a project
    – Or even a new language
  • Hides a lot of (messy) internals
  • Avoids baggage
    – Review process
    – CI / testing infrastructure
    – Release cadence

(these things are good, but intimidating)
**DO**

- New coders will need as much hand-holding as you can provide
- Make it easy to start
  - Get code in front of them
    - Preferably nicely abstracted using APIs
  - Lots of examples
    - UI changes
    - Adding attributes
    - API extensions

Have a template
DON’T Make it hard to start

• New coders want to develop on their sandbox
  – Not a true `development setup`
  – Really a production install

• Quick to set up
  – Ideally a single `git clone` or similar

• Alternatively, write generators instead
  – Highly language/framework dependant
DO  Have good documentation

• Again, hand-holding
• Document what is possible
  – You won’t get *everything* into the template
• Code examples
  – Lots of them
  – More than that 😊
• Limitations
  – Avoid the `gray area`
DO

Provide internal APIs

• Provide `extension points`
  - Some languages will mandate this anyway
  - Others (e.g. Ruby) are more flexible
    • See `alias_method_chain` for how this can be **bad**

• Plugin authors will find a way
  - Nasty hacks
  - Breaks at a later release of core
  - Plugin authors get upset
  - Users blame core, not the plugin

• Ideally, core uses the same APIs
DON’T Break your promises (APIs)

• Once you have APIs, honour them
  – Don’t change them on a whim
• Deprecation strategy
  – Deprecated in X+1 (or more)
  – Removed in X+2
• Feedback loop to authors is key
  – If they don’t know about the deprecation
    then they’ll have to fix it in a rush at release
  – More hackery 😞
THINK

Succession planning

● What happens when an author moves on?

● Code
  - GitHub / GitLab / etc
  - Commit access? Forks?
  - Host it in your organisation

● Packaging
  - Release access
    - RubyGems, PyPi, CPAN, etc
  - Have a generic “community account”
    - Generic account that can be added to any plugin
Quality / Curation

- How good is a plugin?
- How can you ensure plugin quality?
  - Should you care? Silos can work
  - Curation strategy
- Help authors out where possible
  - Allow them to join in the `baggage`
    - at their own pace, as they gain confidence
  - Hosting of bug tracker / code / docs / etc
  - Participation in your CI system
  - Release cadence
THINK

Installation methods

- How do users get the plugin?
  - Not a development question, but important
- This is hard to change later
- Some options
  - Tarball (PHP apps, usually, e.g. NextCloud)
    - Also common for desktop apps
  - Git clone (often in containers, e.g. Discourse)
  - OS Packages (RPM/DEB/etc, e.g. Foreman)
  - ?
- Affects other options, such as discoverability
Keep communicating

• Probably the hardest part
• This is not a `once-and-done` task
• Core will evolve
  – Plugin authors won’t always notice the changes
  – Documentation needs to keep pace
    • Especially all those examples
• Good communication tools helps
  – Users can get help from the right people
  – Devs can keep each other up to date
  – Come to my talk tomorrow about comms tooling! 😊
Key Take-aways
Focus on the barriers
“You don’t see the lost opportunities” - KK
Good communication channels to authors

Thanks!
Questions?
https://community.theforeman.org