CKAN 3.0 Roadmap

Summary

After talking to members of the tech team, members of the Steering Group, CKAN devs and maintainers outside the tech team, and representatives of organizations using CKAN, and after sorting through open issues and older collections of ideas, I ended up with a diverse list of needs, desires, comments and concerns.

Considerations and Goals

Size of Core Team and Achievability

One thing that was only addressed by a couple of respondents (and which was conspicuous in its absence from the responses of those with the grandest visions) was the small size of core team, the slow pace of core development (because there's a lot to cover and not a lot of resources devoted to it), the amount of work that still needs to be done on some ongoing projects, and the uncertainty that the necessary resources will be provided in the future.

This is one of the most important things to consider when drawing the roadmap. We need to be clear and honest when detailing the work that needs to be done (for instance, I think that completely replacing Pylons with Flask in CKAN core and empowering the community to upgrade their own extensions as easily as possible is a bigger project with more tasks than we realize). We need a plan that commits us to completing the boring parts before we get too deep into the improvements. We need a plan that a small core team can execute even if we don't receive significant support. And we need a plan that encourages significant support by not just adopting some of the improvements requested by other stakeholders but by also making them dependent on completing the boring work (working on the latter requires completing the former) as much as possible.

Laying a Foundation

The second major theme is laying the foundation for bigger, better (non-breaking) improvements in future 3.x point releases. There's a lot of work that's worth doing that I think we just won't have the time or resources to do. So we should focus on a 3.0 that lays the groundwork for us to do it in core, or for others to do it via plugins, in the future. This might mean that some of the improvements are more developer-centric or backend-oriented. And it might seem like that makes 3.0 less interesting. But across the board, from developers to stakeholders to users, there was interest in and support for features like adding scheming to core and enabling plugin extras. In some cases the support might be a bit indirect—for example, a user doesn't necessarily know anything about plugin extras, but they do want features that would require plugin extras to exist.

Getting Support and Resources

The third theme is encouraging support from organizations that can devote resources or funding to development. This theme builds on the first two. If we want to build the really flashy features, we'll need help building the foundation. And so we'll need to motivate vendors or users or other organizations to help us so that they can get the features that they want. And since we don't have the resources to build everything that we or they want, we should focus on providing them with features that are both beneficial to them and that they can (or even better, must) build on. In addition to scheming and plugin extras, which fit this criteria, a feature like opt-in "phone home" for keeping track of instances, plugins, etc. is something that vendors want and would provide support for and at the same time it's something that can serve as the foundation of a more interesting feature, like WordPress-style plugin management and an improved extensions.ckan.org.

Usability

Fourth is usability, for devs, users, and admins alike. Again, some of the developer-centric or foundational changes are also going to bring about improvements for other groups, so even if we can't improve the user experience ourselves in some cases, we can still say that we're making it possible for extension developers and vendors to improve UI/UX thanks to our work.

Trimming

Finally, there's trimming the code base. While everyone doesn't necessarily agree on what to cut, everyone seems to agree that CKAN needs a trim in order to grow. The DataProxy, stats and tracking, etc.—there are some good candidates. We're already going to be making breaking changes and we have limited resources. I think, as we move ahead, we'll need to conserve our energy. If we just don't update X, we'll have more time to devote to improving or adding Y. It may be more helpful to think not of cutting a certain feature but rather of not carrying it along into 3.0.

There are some things we should just leave behind. We should be on the lookout for them as we work.

Conclusions

First we need to modernize CKAN and place it on sound footing for the future. Some of these items are ongoing work—I don't think that they'll be truly finished anytime soon, so I think it makes sense to consider them part of 3.0. And: I think we need to take the extensions maintained by the CKAN org into account when planning. As well as documentation. That means something like transitioning to Flask is only finished once the CKAN org can say that you can deploy Flask CKAN and that our own extensions work (because who uses CKAN without any extensions?) and that we can guide users step-by-step through the process of updating their own extensions.

Roadmap

This roadmap lists some things as individual items that others might group (or hide) within a single item. The intent is that we don't lose sight of how big some tasks are.

1. Make CKAN core Flask-only.
    i. Plan and communicate a transition period so that plugin authors have time to update their own plugins.
    ii. Remove all traces of Pylons.
    iii. Remove Paster and replace with a Click-based CLI.
2. Select extensions hosted under the CKAN GitHub organization that will be supported for version 3+ and make them Flask-only.
    i. And replace Paster with Click.
3. Replace VDM (among other issues, it is blocking Python 3).
    i. Activity Stream improvements are nearly finished, for detailed records of dataset changes
    ii. Then Activity Stream detail needs extending to group/org and member.
    iii. Then we can remove VDM and all revision tables
4. Make CKAN development Python-3 focused. This depends on the preceding items and on dependencies that are holding up pull requests.
5. Select CKAN organization plugins that will be supported for version 3+ and make them Python 3-focused.
6. Create a Pylons-to-Flask transition guide for core developers.
7. Create a Pylons-to-Flask transition guide for extension developers.
8. Create a Paster-to-Click transition guide for extension developers.

9. Update the docs in general to resolve inconsistencies and make on boarding easier.
    i. For those looking to install and use CKAN, but not to develop it.
    ii. For developers.
10. Switch from Nose to Pytest.
    i. Remove legacy tests: Verify that the modern tests cover everything that the legacy tests cover and add modern tests for anything that's covered in the legacy tests but not covered in the modern tests.
11. Upgrade from Bootstrap 3 to Bootstrap 4.

And we need to improve CKAN and empower others to improve it even more:

1. Integrate ckanext-scheming in core.
    i. And update all the docs, tutorials, etc.—don't just bundle it, but make sure everyone is taking full advantage of it (e.g., more attention to scheming's public API for publishing schemas.
    ii. Add dynamic Solr schema support to enable scheming in core.
2. Better data importing to the DataStore, with data description via schemas.
3. Enable plugin extras.
4. Opt-in "Call-home" service to check for new updates and warn about vulnerabilities.
5. Pluggable search engine: full support for Solr and Elasticsearch.
    i. This will also be the start of improving the search experience.
6. Spatial metadata/search in core (both Solr and Elasticsearch support searching WKT or GeoJSON fields)
    i. We can make spatial search better (they support more than just bounding box queries) and signal that we're moving to support geospatial data in other ways in the future).
7. Customizable activity streams / complete replacement of the old revisions.
    i. And done in a way that satisfies all the users who need to be able to track revisions.
8. Support Topics/Themes out of the box (it seems like everyone tries to turn Groups into Topics).
9. Expanded roles (like Contributor) and finer-grained identity management and permissioning.
10. Pluggable themes (users can activate a range of themes via the admin panel, maybe even install and activate them from the CKAN site).
    i. If possible, expand this to a full-fledged plugin management console enabling devs and admins to search, install, configure, and update extensions and plugins easily.
11. A simplified Docker workflow that enables one to easily stand up a CKAN portal, complete with database and search.