CKAN Cloud RFP Draft

General

The following are general points that all solutions should comply with:

- **Open-source**: all delivered software will be open-source. Specifically, it must be dual-licensed under the MIT and AGPL
- **Tested**: all delivered software must have appropriate unit and/or functional tests. Tests should have reasonable coverage (not necessarily complete but substantial and sufficient for others to validate all major functionality and to continue development in future). Tests must be passing. Where code will be contributed to the CKAN core codebase or to a CKAN extension tests should follow standard practice.
- **Accepted into Core**: Code (and associated tests) for core CKAN should be accepted by the CKAN technical team as of appropriate quality and should be scheduled for the next appropriate release
- **Agile and publicly accessible**: Development should proceed, where possible, in an agile manner and in publicly accessible repositories

User Stories

Personas:

- **Cloud Admin** (also multi-tenant admin)
    - A manager or administrator of a cloud environment (someone who creates or shuts down new CKAN instances)
    - Note: same as Admin for Multi-tenant
- **Instance Admin**
    - Administrator of a CKAN instance (site) e.g. someone who edits config, adds users etc
- **CKAN Developer** - someone developing CKAN related functionality
- **User** - user of a CKAN instance (authenticated - perhaps an organisation manager …)

1 Enhancements to Core CKAN

These user stories are about enhancing core CKAN that will make it easier to to deploy and manage multiple instances.

For example, changes so that a CKAN instance can be administered easily by an Instance Admin without requiring high-level sysadmin support (e.g. someone who has access at the command line level to the machine the instance is running on). This is important because the owner of an instance should be able to administer and configure that instance without requesting manual changes by the Cloud Admin (e.g. one wants to avoid an instance owner contacting the cloud admin every time they want the site title changed)

US 1.0 User Sessions Stored in Database by Default

As an Instance Admin I want user sessions to be stored in the Database and for session length to be defaulted to a reasonable length (e.g. 1d or 7d) so that I don't have to worry about sessions on disk using too much disk space and so that sessions are better encapsulated

- This should just be a config option but may need checking and may need some discussion as to whether to make this the default for CKAN generally

US 1.1 Manage Config (Settings) via Web Interface

1.1.1 As an Instance Admin I want to set config (such as site title and as many other possible config values - see http://docs.ckan.org/en/latest/maintaining/configuration.html) from a web interface so that I can change these settings without having to use the command line (and without needing access to the machine the instance is running on)

1.1.1 As a Developer I want to set config from a web API so that I can change these settings from my program (or from a UI that uses AJAX)

Notes:
Implementing this will likely involve following pieces of work

1. Moving most configuration to be in Database (in a new table) and making relevant changes to the app in light of this (e.g. so that app loads config from the DB)
    a. Ensuring that changes in config result in a reload of this config in the app (e.g. by changing app to load DB config on every request)
2. Maintaining backwards compatibility by still loading config from disk for cases where some or all of config is still on disk (so search disk first, then load DB config - DB config overwrites disk config)
3. Create a Web UI as part of CKAN Admin Dashboard for viewing and updating config (e.g. located at ckan-admin/config)
    a. See US#2.2 below for changes that would support this

Config in the Database

- All config should move to the DB except for that which cannot e.g. DB connection information (cf wordpress)
- Suggestions re new config / settings table schema:
    - id, namespace, name, title, value, description, help, value_type, dictname
    - value_type is an html5 types (?) - e.g. text, textarea, integer, decimal, email, date etc … - used to help display info in text interface and validate.
    - dictname their for backwards compatibility to be a place to store current name in pylons config dictionary
    - namespace + name are unique
    - Translation algorithm to go from db entry to entry in ckan config dictionary: if dictname exists use that. Else: concatenate namespace with '.' and name e.g. namespace = 'ckanext.stats', name = 'cache' becomes 'ckanext.stats.cache'
    - Note: this means that plugins can write to the same table
- namespace defaults to 'core' which is also '' (empty string)
- Note: we will need a default set of settings (including help text etc). This should probably be maintained as a CSV or JSON file (and move out of ini) and then this is used to populate the database on boot.
    - all CKAN config settings along with their defaults, help texts, whether they are required, validation, etc to be defined in one place (this will also enable auto generating the config settings reference docs from that source).

US1.2 Review current CKAN Admin dashboard and enhance as appropriate

As an Instance Admin I want to administrate my CKAN instance (e.g. change config, add/remove users, set permissions) from a web user interface so that it is easy for non-geeky and users

without sysadmin access to the underlying machine to administrate the instance

- Review current CKAN Admin Dashboard (currently very limited in extent)
    - Suggestion: Model on Wordpress Dashboard
- Tidy and/or improve if needed
- Point the way (but not necessarily implement) support for things like user administration (cf #XX)
- [Nice to have] Add support for plugins/extensions to extend the dashboard
    - This allows extensions to create their own admin interfaces
    - Question: is this needed or is controller support that we have good enough?
        - Ans: probably want more than this. Whether it be helper functions or even if just documentation of how to do this

US1.3 Multi-tenant CKAN [Very Optional]

As a Cloud Admin (or general Sysadmin of CKAN) I want to run multiple CKAN instances off the same codebase so that I can manage it more easily

Note: this is related to but somewhat parallel to a direct approach to booting multiple CKANs - the simplest approach is simple to install many sets of CKAN code (e.g. using Docker). However, multi-tenant offers quite a few advantages (e.g. maintaining one codebase, or sharing user data across instances). As such this user story is optional.

Implementation details:

- Likely setup is a single CKAN (code) application "instance" which serves many site "instances" by switching config based on URL and therefore serve different sites (with different data etc)
    - A single code-base and single active "instance" serving multiple distinct customers
- Qu: do we share the Database across all the instances?
    - Could go either way. Initial feeling is that you don't need to share DB but share code
    - Share DB would involve prefixing all tables with a configurable prefix (cf wordpress)
- Qu: how do we handle plugin activating per instance (this is not about installing the plugin which is obviously across all instances)
    - ANS: that should be in the config DB so not a problem
- Qu: do we share user database across the multi-tenant instances?
    - That would require a shared database and would require changes to schema e.g. to indicate per instance what users were "live" on that instance
    - Suggestion: do not support this.

US 1.4 CKAN Setup on First Boot Wizard [Optional]

This user story is considered as an optional "nice to have".

As an Instance Owner I want to be walked through setting up my CKAN Instance on first use (including basics such as creating an admin account) so that my instance is configured and ready to use

- Idea is something similar to wordpress experience once the basic config (on disk) such as DB connection is done.
- At its simplest just a prompt to set admin account (if no user account yet exists) and perhaps a walkthrough of setting site title etc
    - e.g. New "Admin" Visits CKAN instance homepage and is prompted to set admin password and is taken through any further info
    - [One could (maybe) support even setting DB connection info but that would involve web server user having write access to base config on disk which may not be a great idea for security reasons]

US 1.5 Tracking API usage and other usage like disk space [Optional]

This is useful to Cloud Admin as measuring usage across different instances may become important to performance management.

Tracking API usage and other usage like disk space (and displaying this in admin interface) [#72]

1.5.1 As an Instance Admin I want to monitor how much usage is being made so that I can manage my usage and plan (e.g. around charging or performance)

1.5.2 As a CKAN User I want to know how much use is being made of my data so that I can report this to other people and manage my usage (or others' usage) of the platform

US 1.6 User Administration in Admin Interface

[Optional] As an Instance Admin I want to add (invite), edit, and remove users so that they have access (or do not have access) to the platform

[Required] As an Instance Admin I want to make a user a sysadmin in the user interface so that they have full access (and I don't to have direct access to the machine to do this …)

2 Making CKAN Fully "Cloud Ready"

These user stories are primarily about supporting infrastructure that make deploying and managing many CKAN instances efficient, easy and scalable on relevant PaaS platforms such as AWS.

2.1 Dockerized CKAN

As a Cloud Admin (or any kind of CKAN Admin) I want to deploy CKAN in a docker container easily so that I can run CKAN in docker (and hence on a relevant PaaS)

- Dockerize CKAN core
- Dockerization of associated things like DB is desirable but may be optional. Key thing is that connection to e.g. DB etc must be easily configurable into CKAN docker container
    - Optional as DB may not want to be docker deployed e.g. if using AWS you may want DB in Amazon RDS
- *Note: quite a bit of work on a dockerized CKAN has been done but review will be needed of existing work plus any relevant improvements - https://github.com/ckan/ckan-docker*

2.2 Automated deployment and initialization of new CKAN instances on a PaaS (e.g. AWS)

2.2.1 As a Cloud Admin I want to create (and manage) a CKAN instance (site) in my cloud farm so that it is live and available online at a URL

This user story is really a very high-level user story and there are a lot of smaller ones corresponding to key desired activities such as

- Create and remove a farm "environment" (e.g. DB server, VPN etc)
- Create an instance within that environment (installed and ready but not live online)
- Activate an instance (make it live online)
    - Setup any associated monitoring
- De-activate (take it offline but don't destoy it)
- Purge (Destroy it plus all data - perhaps with backup)
- Plugin install, activate, deactivate, deinstall (per instance (?))

Implementation notes:

- Support must be provided for one or more major PaaS such as AWS or OpenStack
- This process should be fully automated - so e.g. booting a new instance should be one command on the command line or a click of a button
- This functionality should be wrapped in a python library so that it can be used to power a web application or similar (see later user stories)
- Relevant information arising from all these operations (e.g. details of the farm, details of instances must be persisted)
    - Details are not fully determined and left to implementor (and will intersect with other later user stories e.g. re creating UIs). Suggestion is that config either be simple JSON or a basic DB
- Bonus: nice UI for launching and monitoring (see next item)

3 Web-Based Workflow for Creating and Managing Cloud Instances

These user stories are about "wrapping" the work of the previous section in user-friendly interfaces and workflows.

3.1 Request / Create CKAN Instance (Potential Instance Admin)

As a (potential) Instance Admin who wants a new CKAN site I want to have a simple web-based interface for requesting a new instance (and having instance rapidly booted in response) so that I have a new CKAN site quickly and easily

Notes

- Simple form where you can fill in your details (e.g. Name, Org, Email, Password)
- Notified in response to submission and/or by email (soon after) that new instance is ready and here is the url
- Go to that instance and go through any next steps (cf US#1.4)
    - Note US#1.4 could be *less* relevant if relevant info captured in form here and used in booting the instance …
- User experience should be excellent (as simple and easy to use as possibly but no simpler!)
- Instance should be booted automatically (suggest using the library / scripts produced re US 2.2)
    - Simpler option (less preferred) is email to Cloud Admin who takes some action (see next US 3.2)

3.2 Management Interface for CKAN Cloud Farm

As a Cloud Admin I want to view and administer my farm of CKAN instances in a web UI (or via a Web API) so that I can manage them easily (and have a non-command-line user do this)

- Simple Web API and UI for adminning a cloud CKAN environment
    - e.g. Wrap the code from US 2.2 in simple JSON based Web API
    - Wrap that with simple UI that allows one to admin all CKAN instances in a given environment
    - Upgrade, start, stop, check usage etc
- Single login to UI is OK - e.g. could even use apache/nginx login
    - Do not need multiple users and types of user
    - However, designing so that this is possible longer-term is a nice to have

4 General Activities

US#4.1 Connect with CKAN Association Technical Team

As a Developer building new functionality I want to make my functionality builds off and contributes maximally to enhancing CKAN so I and the community get maximum and sustainable value from what I do

Goal: Ensure this work is well integrated with general CKAN roadmap and includes engagement with the CKAN Association Technical team