RefStack requirements

Introduction

This document is an attempt to clear any discrepancies in understanding of stakeholders and list complete set of requirements in view of newly proposed enhancements: RefStack enhancements document

It serves to be a source for further architectural and design decisions.
The initial set of use-cases/user stories is taken from here: Use Cases
with vocabulary taken from here: Lexicon
New design suggestions are also taken from this presentation: RefStackDataModels

This document is supposed to be a discussion place where we can agree upon the final set of use cases and update the wiki afterwards.

Existing requirements

This section is completely taken from https://wiki.openstack.org/wiki/RefStack/UseCases

There are many questions which arise upon reading the existing requirements. It might be that the best way to proceed would be to rephrase them starting from the most basic stuff and make sure that there is no more ambiguity left. Such an attempt is made further in the document.

Ops: A Private Cloud Operator

I want to:

- **preview results before they are public**
- compare my test runs over time
- compare my test runs against clouds with similar characteristics (tagging)
- **be able to ignore tests that are not important**
- **have tests grouped into capabilities so I am not overwhelmed**
- **highlight tests that are critical for operations**
- **be able to upload my tempest results to refstack for analysis**
- **be able to remove results**

User: An OpenStack User

I want to:

- make sure that the capabilities I require are available
- op 3, 4, 5 and 6
- **be able to post results from my own tests**
- (foundation 7)

Prospect: a Prospective OpenStack User

I want to:

- compare results without uploading tests
- not be forced to create a lp account to get data

- not see results that are not accurate (approved results only)

Vendor / Organization: an OpenStack Vendor

I want to:

- control tags that can be applied to test run against my cloud
- contest results that I disagree with
- **be able to mark a result as official**
- **know that test submission are coming from validated users**
- be able to excludes tests for capabilities we have not implemented (ops 4)
- be able to give customers information about my compatibility (ops 3)
- results that I don't control to be statistically relevant
- expose aggregate results to prospects (if they are favorable)

Foundation: the OpenStack Foundation

I need/want:

- **way to determine which tests are passing in my install base**
- **way for community to indicate important capabilities**
- way to communicate which tests are "must pass" on a version by version basis
- way to certify that a result includes all the "must pass" tests
- to indicate organizations that are in compliance
- way to settle disputes (ops 3)
- to highlight gaps in capability coverage
- to ensure that results posted are accurate
- link between RefStack result and documentation at the appropriate version

Vendor: Relies on OpenStack API's

I need to:

- be able to validate which clouds my product will work with (user 1)
- be able to upload test results that are outside the upstream

Terminology and entities

The Terminology is based on the vocabulary defined by DefCore here:
https://github.com/openstack/defcore/blob/master/doc/source/process/Lexicon.rst

Some terms are adapted to technical and architectural means of RefStack but their meaning here should complement, not conflict the DefCore-defined meaning.

Actors

- **Foundation** - OpenStack Foundation
    - Produces OpenStack platform
    - Defines DefCore Guidelines and Schemas
    - Implements core Tests
- **Vendor** - Vendor / Organization
    - May manufacture a Product based on OpenStack platform (Mirantis, IBM, proprietary OpenStack deployment)
    - May manufacture non-OpenStack cloud platform (Amazon, Google)
    - May develop a client cloud Product deployed into an OpenStack or non-OpenStack based Cloud (Pivotal with its Cloud Foundry)
- **Provider** - An Operator of Cloud
    - Approved by Foundation official Cloud Operator. Operates an OpenStack or non-OpenStack cloud manufactured by one of the Vendors (may be in both roles Vendor and Operator simultaneously in case of proprietary solution)
- **User** - Regular OpenStack (non-OpenStack) User
    - May use some Product
    - Has identity in a form of OpenID
    - May be unofficial Provider or Vendor
- **Guest** - Prospective OpenStack (non-OpenStack) User
    - Doesn't have any identity - anonymous access

Entities

- **Cloud** - any OpenStack or non-OpenStack cloud
- **Test suite** - a set of Tempest tests from core Tempest or provided by plugins
- **Guideline** - a set of rules, expressed by tests, their grouping into components, capabilities and target programs, used for validating clouds. DefCore Guidelines are defined by DefCore committee for OpenStack certification
- **Schema** - a json file containing Guideline of particular Version
- **Capability** - The functionality ensured by a set of tests collected into a group
- **Component** - A collection of functionality generally used together (e.g.: object, compute) covering a number of Capabilities
- **Target Program** - a set of components validating a Cloud to comply with some particular part of a Product's functionality (Compute, Object, EC2 API)

- **Product** - an OpenStack (OpenStack itself) or non-OpenStack (Amazon) Cloud solution, used to create private or public Cloud, commercially sold (Mirantis, Piston) or custom-made (private OpenStack installations); also a Cloud-based solution which is supposed to work in a Cloud (Pivotal Cloud Foundry)
- **Test** - a single test that exercises functionality of a Component to validate expected behavior and provides pass or fail judgement
- **Test run** - a process of running some Test suite for validation of some Cloud which produces a set of individual test results and have a cumulative Fail/Pass status
- **Test results** - full set of results from a Test run
- **Group** - a Group of users
- **Owner** - admin of an Entity (probably one of the admins - particular mechanism is out of scope here)
- **Creator** - User which created an Entity
- **CloudID** - A unique identifier which can be used to definitely determine a Cloud; it should be possible to be generated automatically when tests are run on premises

Functional requirements

This section provides an attempt to restructure the existing requirements, trying to provide more details and eliminate ambiguity.
Most probably some unclear requirements won't be reflected here at the beginning.

Tagging functionality is deliberately omitted here because it is possible that it might not be required now for the same motives that were valid earlier - new entities and their relationship might provide required functionality.

Please read the whole section first, some questions might be answered in the subsequent scenarios.

Scenarios

Initially it's presumed that Actors' permissions increase bottom-up from Guest to Foundation. So further Actors have all the use-cases listed in previous Actors.

"his/her" in the text below concerns all of the users in the group managing some particular object (Provider removes his/her Cloud - any user in the Provider's built-in admins group can do this).

Guest

1. Guest browses public objects and information
    a. Example: Guest browses existing officially registered Clouds (Internapp, Walmart private cloud, Amazon, Google)
    b. Example: Guest browses existing officially registered Products (Mirantis OpenStack, Piston OpenStack, EC2 API, Pivotal Cloud Foundry)
    c. Example: Guest browses unofficial Products and Clouds
2. Guest browses results of available validations and can tell which are successful and which are not
3. Guest browses the results and can tell which User/Cloud/Product/Vendor they are related
4. Guest browses Clouds compliant with particular Product/Target Program.

User

5. User validates (runs tests) his/her Cloud for compliance with particular Target Program of particular Version
    a. Test results are produced and stored with UserID (or anonymous) as Creator and CloudID
    b. If no Cloud with such CloudID exists it is created with this UserID as Owner and Creator (if not anonymous)
6. User browses results of his/her validations

7. User removes his/her results
    a. Test results are removed
8. User registers his/her Cloud supplying the CloudID
    a. Cloud is stored with new CloudID if no such Cloud exists
    b. User becomes an Owner and Creator of existing Cloud with such CloudID if it's orphaned (was created before by means of anonymous test results and has no Owner)
9. User removes his/her Cloud
    a. Cloud is removed
    b. Test results are either orphaned or removed accordingly User's choice
10. User associates/disassociates his/her results for some unregistered Cloud with some of his/her registered Clouds
    a. Cloud gets one more CloudID associated
11. User registers a Provider with the Foundation
    a. Any User can submit an application to register a Provider
    b. Such a User becomes an Owner, "PTL" and Creator of such a Provider if registration is approved
12. User registers a Vendor with the Foundation
    a. Any User can submit an application to register a Vendor
    b. Such a User becomes Owner, "PTL" and Creator of such a Vendor if registration is approved
13. User registers his/her Product
    a. New private Product is created with the User as the Owner and Creator
14. User registers his/her Schema(s) with Guidelines for validation of particular Product (optional)
    a. New private Schema is created with the User as the Owner and Creator
    b. The Schema gets associated with the Product if specified
15. User associates his/her Schema with particular Product if it was not associated before
16. User submits his/her Schemas to Vendor of some Product in order to publish them
17. User deletes his/her Schema(s) if not owned by Vendor yet
    a. All the Test results for such Schema(s) are removed
18. User runs tests from a command line utility on his/her or Cloud's premises and results are automatically or manually uploaded to RefStack
19. User runs tests from the RefStack UI for the Clouds allowing external access sufficient for testing
20. User associates his/her Cloud with particular Product(s)
21. User makes his/her Test results public
    a. They are not "official" and are available only for viewing (not used for official validity/certification results for any object) until they are approved by Provider

Provider Admin (Provider)

22. Provider makes results of his/her validations public
    a. He/she is suggested to make his/her Cloud public if it's not
23. Provider makes his/her Cloud public

    a. He/she is suggested to make results for this Cloud public
24. Provider manages Group of Users allowed to control Clouds belonging to this Provider and Provider itself
25. Provider views and approves someone else's Test results for his/her Cloud and makes them official and relevant for his/her Cloud

Vendor Admin (Vendor)

26. Vendor makes his/her Product public
27. Vendor makes his/her Schemas for the Product public
28. Vendor approves and makes public Schemas submitted by some User
    a. Vendor becomes an Owner of such Schemas
29. Vendor manages Group of Users allowed to control Products and Schemas belonging to this Vendor and the Vendor itself
30. Vendor deletes his/her Schema(s)
    a. All the Test results for such Schema(s) are removed

Foundation Admin (Foundation)

31. Foundation approves application for Provider registration
    a. New Provider is registered with new ProviderID
    b. User applied for registration becomes an Owner
    c. Clouds owned by this User get associated with the new Provider
32. Foundation approves application for Vendor registration
    a. New Provider is registered with new VendorID
    b. User applied for registration becomes an Owner
    c. Clouds owned by this User get associated with the new Provider
33. Foundation registers itself as a Vendor "OpenStack Foundation" for Product "OpenStack" and serves as a Vendor for this product henceforward
34. Foundation manages Group of Users allowed to control everything (admins)

Non-functional requirements

Usability

1. Allow grouping of tests for better comprehension and presentation (capabilities)
2. Highlight tests that are critical for operations
3. Be able to ignore tests that are not important
4. Make sure that required capabilities are available
5. Lists of data presentation should be fast and convenient (paging, grouping, filtering)
6. It should be clear which Entities belong to what User and/or Organization

Security

7. RefStack shouldn't store permanently (for more than a span of running one test session) any confidential or potentially confidential information
8. RefStack shouldn't expose temporary confidential information coming in data available for browsing (logs, configs)

Availability

9. Testing process should not affect neither results of such testing, nor other testing processes
10. Stored testing results should not affect other testing results or storage
11. Testing time shouldn't exceed 1 day after which the process should be stopped
12. RefStack upgrade procedure should preserve all the data and take no more than 2 hours of downtime for the services

Performance

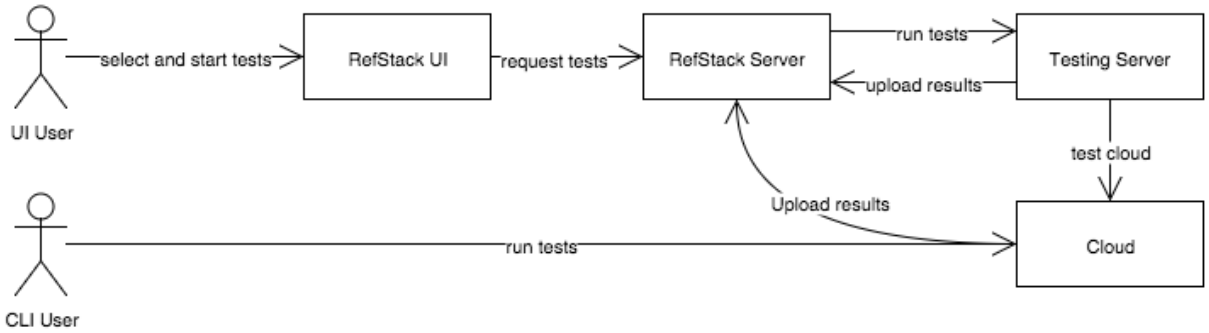
13. Response time from any action of RefStack UI shouldn't exceed 5 seconds. Long actions like running tests should be performed asynchronously and report their progress in some way

Architecture

The following is not supposed to be here in the Requirements document. It is temporarily placed here for stakeholders to see the proposed design. It is by no means complete now.

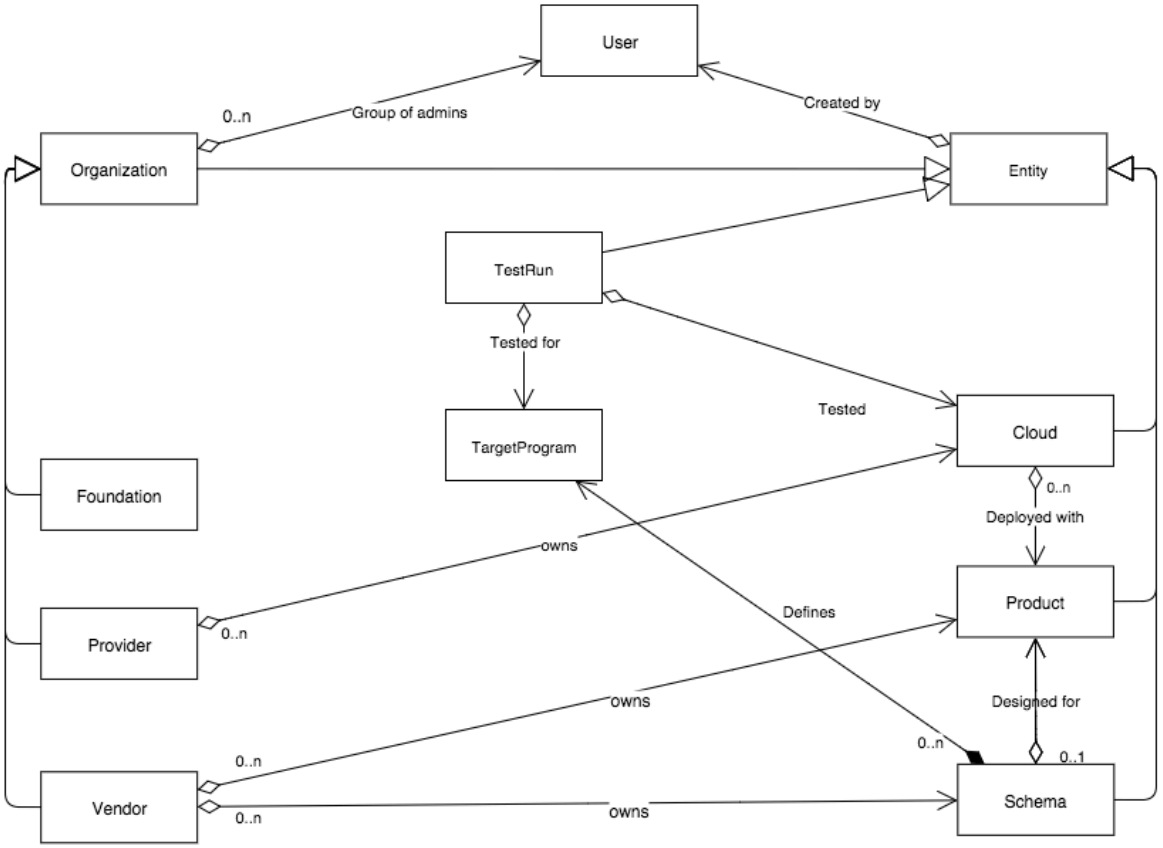
Diagrams are done in the UML notation.

Testing options



Testing can be done either via RefStack UI (centralized testing) or via RefStack Client (CLI utility).
In the first case everything is done through UI, but the tested Cloud should be accessible to the internet in an extent sufficient for the testing (usually it is just public REST APIs)
In the second case User runs RefStack Client CLI utility on some machine which has sufficient access to the tested Cloud and results are automatically or manually uploaded to RefStack Server.

Domain model



- Organization/Cloud/Product/Schema/TestRun are Entities having a User who created them
- Vendor/Provider/Foundation are Organizations which are controlled by a group of admin Users (please note that no explicit Group is required in such an approach - it is suggested that intrusive list of admins in Organization is sufficient for existing requirements)
- Provider owns and controls a number of Clouds
- Vendor owns and controls a number of Products with their Schemas
- Schema consists of and defines several Target Programs which are used in Test Runs to test a Cloud
- Schema is associated or not with one Product
- A number of Products can be associated with any particular Cloud

Roles

No explicit Roles or role management is required.
Implicit roles are:

- Admins - all members of the Foundation built-in Group of Users
    - Can do anything
- Vendor Admins - all members of the particular Vendor built-in Group of Users

    - Can manage Vendor, its Users, Products and Schemas
- Provider Admins - all members of the particular Provider built-in Group of Users
    - Can manage Provider, its Users, Clouds and Test results
- User - regular User
    - Can manage his own objects
- Guest - anonymous User
    - Read-only access to UI and public information and objects

Schema

Schema is a set of Guidelines designed by DefCore and expressed in JSON format. With introduction of external Products and Test suites it is suggested to allow other Vendors to provide their Schemas and extend the format of the Schema to allow storing of some additional information.

The following changes are suggested:

- Target Program - the same as it is now in RefStack: a set of components and tests used to validate cloud for some particular role
    - Target programs and their relationship will be defined in Schemas (not hardcoded in code as it is now)
    - A Badge can be associated with each Target Program - it'll allow results of validation to be displayed in visual details showing which Target Programs a cloud satisfies (OpenStack logo is one of such badges).