Polytrix: A Pacto-Powered Polyglot Test Matrix

Max Lincoln and Fernando Alves

ThoughtWorks Brazil
Av. Gov. Agamenon Magalhães, 4779, 12º andar
Empresarial Isaac Newton
Ilha do Leite, Recife - PE
50070 160 - Brazil

Abstract. We have created a polyglot test framework named Polytrix to compare, benchmark, and independently verify a suite of open-source OpenStack SDKs that each target a different programming language. The framework validates sample code from each SDK against a shared test scenario to validate that each SDK correctly implements a given scenario. It uses Pacto for integration contract testing between the SDKs and the OpenStack services, and generates test reports that help compare and document each SDK. It is designed so interactive training materials can be generated in future versions.

1 Introduction

OpenStack was founded by Rackspace Hosting and NASA in July 2011 as an open-source system for building public and private clouds[1]. Since then, it has grown into a global open-source initiative, with contributions from more than 1800 engineers and 168 companies[2], and users in more than 132 countries[3]. Rackspace has continued to be an major contributor to OpenStack throughout the latest (Havana) release[4], and is also a major contributor to many projects that have grown around OpenStack. These other projects include OpenStack SDKs. These SDKs fall into two groups[5]:

OpenStack SDKs: only support clouds built with OpenStack (including OpenStack private clouds as well as public clouds from Rackspace, HP, and IBM)

Multi-cloud SDKs: support OpenStack in addition to other clouds

OpenStack has not officially selected supported SDKs, but Rackspace supports OpenStack SDKs written in .net (openstack.net), php (php-opencloud), python (pyrax) and go (gophercloud) as well as multi-cloud SDKs written in java (Apache jclouds), node.js (pkgcloud) and ruby (fog)[6]. The three multi-cloud SDKs are each governed by a different organization.

The OpenStack wiki does make the requirements of an SDK clear:

1. A set of language bindings that provide a language-level API for accessing OpenStack in a manner consistent with language standards.
2. A Getting Started document that shows how to use the API to access OpenStack powered clouds.

L. Corral et al. (Eds.): OSS 2014, IFIP AICT 427, pp. 191–194, 2014.

3. Detailed API reference documentation.
4. Tested sample code that you can use as a "starter kit" for your own OpenStack applications.
5. SDKs treat OpenStack as a blackbox and only interact with the REST/HTTP API.
6. Must be sustainable.
7. License must be compatible with Apache License v2.

The quality and completeness of SDKs can be difficult to assess, especially as the number of SDKs grows. The OpenStack wiki states: "What constitutes an official OpenStack SDK has not been determined. This is an area the needs more work."

This paper outlines a test framework and documentation generator codenamed Polytrix (Polyglot Testing Matrix) that assesses SDKs and produces documentation that may help them achieve officially supported status.

2 Solution Overview

Polytrix[7] was using Ruby's RSpec[8] testing language, the Pacto[9] integration contract testing library, and the Docco[10] documentation generator. Since we are testing many different languages, we needed a generic way to update dependencies before running tests. We used the GitHub 'script/bootstrap' pattern[11] to keep things language agnostic. Once each project was bootstrapped, the Polytrix invokes each test. We run a Pacto server in the background to act as a test stub, mock or spy[12], capturing and validating calls made to services against their expectations that are defined in files referred to as "Pacto contracts".

Figure 1 shows the flow of a Polytrix test. We first launch Pacto as a server so it can accept requests over HTTP from the SDKs (step 1). Then we locate (step 2) and execute (step 3) the code sample for each SDK. The test environment is configured so calls from the SDKs to OpenStack are proxied through Pacto (step 4). Pacto will validate each request and forward it to the real service (step 5), or could return a stubbed response for fast feedback. Validation includes checking the structure of each request and response against json-schema[13] defined in the "Pacto Contract", as well as checking for expected HTTP headers. Once the sample code finishes executing, we check that it executed without errors (step 6). We then compare our test expectations against the actual interactions observed by Pacto, making sure that the expected services were called with appropriate data (step 7). If necessary, we could query OpenStack to check the final state (step 8), but step 7 is usually sufficient. Finally, we instruct Pacto to teardown any resources created during testing by sending deletion requests to counteract any creation requests that were detected during the test (step 8).

The role that Pacto fills is similar to that of the CC helper used by Codecademy's Submission Correctness Tests. The primary difference is that the CC helper validates locally invoked functions, why Pacto validates the invocation of remote services. This similarity makes it possible to enhance the test suite to work as an interactive online programming course for the SDK, similar to how CS Circles[14] or Codecademy teach general programming.

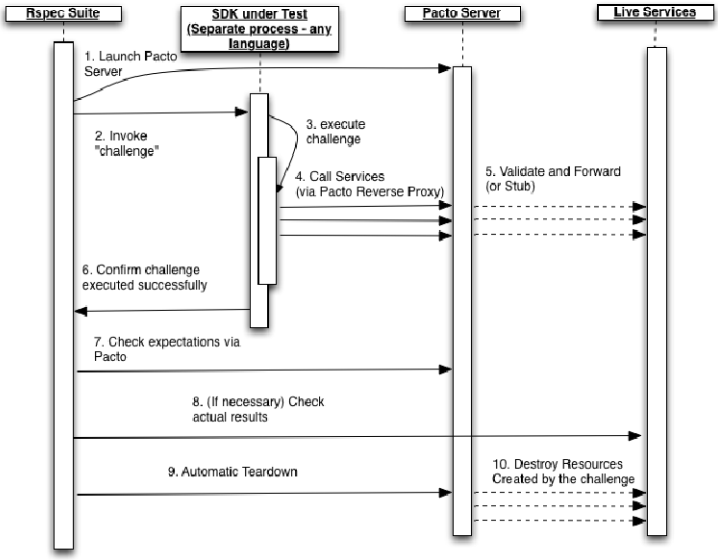

Fig. 1. Polytrix Testing Sequence

Once the test is complete, we generate documentation from the test metadata and extract additional documentation from the code samples using Docco[10], a tool inspired by Literate Programming[15].

3 Tutorial and Future Work

In the tutorial we will demonstrate how Polytrix can help projects like OpenStack assess multiple SDKs by:

- Collecting and displaying sample code for common scenarios in a central location so that the completeness and consistency with language standards of language-level bindings can be reviewed.
- Testing code samples, including verifying that they interact with as expect with REST/HTTP APIs.
- Generating Getting Started Guides and Starter Kit documentation from the test suite.

We will also discuss how Polytrix can be adapted to validate user input instead of pre-written code samples in order to create an interactive online programming course for the SDK.

References

1. 'rCloud' Is Your Cloud: The OpenStack Journey - The Official Rackspace Blog. Rackspace Hosting, `http://www.rackspace.com/blog/rcloud-is-your-cloud-the-openstack-journey/`

2. Stackalytics | OpenStack community contribution in all releases, http://stackalytics.com/?release=all (last modified January 31, 2014, accessed January 31, 2014)
3. Home » OpenStack Open Source Cloud Computing Software, http://www.openstack.org/ (last modified January 31, 2014, accessed January 31, 2014)
4. Who Built Havana. OpenStack Summit (2013)(November 8, 2013) (print)
5. SDKs - OpenStack, https://wiki.openstack.org/wiki/SDKs (last modified January 31, 2014, accessed January 31, 2014)
6. Rackspace Developer Center, http://developer.rackspace.com/ (last modified February 02, 2014, accessed February 02, 2014)
7. rackerlabs/polytrix, https://github.com/rackerlabs/polytrix (last modified February 04, 2014)
8. Chelimsky, D.: The RSpec book: behaviour-driven development with RSpec, Cucumber, and Friends, Pragmatic, Lewisville, Tex (2010)
9. thoughtworks/pacto, https://github.com/thoughtworks/pacto (last modified February 03, 2014)
10. jashkenas/docco, https://github.com/jashkenas/docco (last modified February 04, 2014)
11. Bootstrapping consistency, http://wynnnetherland.com/linked/2013012801/bootstrapping-consistency (last modified January 28, 2014)
12. Meszaros, G.: xUnit Test Patterns: Refactoring Test Code. Pearson Education, vol. 1 (2007) ISBN 9780132797467
13. IETF, JSON Schema: core definitions and terminology, draft (2013), http://tools.ietf.org/html/draft-zyp-json-schema-04
14. Pritchard, D., Vasiga, T.: CS Circles: An In-Browser Python Course for Beginners. arXiv:1209.2166 [cs] (2012)
15. Knuth, D.E.: Literate Programming. Comput. J. 27, 97–111 (1984)