Unified API Governance in the New API Economy

by Chandra Krintz and Rich Wolski

MANAGING DIGITAL ASSETS

Digital assets are becoming the value-carrying resources that underlie much of today's economic activity. Increasingly, businesses depend on the ability to produce, manage, trade, and, perhaps most problematically, destroy digital artifacts (software and data) as key components of commercial functionality and profitability. Because these assets exist entirely within computer systems that are interconnected via networks, new techniques for managing them, such as Hadoop, cloud computing,[1] DevOps, and NoSQL, continue to proliferate. At the same time, previously successful software and IT approaches (e.g., service-oriented architecture, Web services, and machine virtualization) are enjoying a renaissance of utility.

Providing software and data as a service — that is, enabling immediate, authenticated, and scalable networked access to digital assets — is critical to the success of any commercial enterprise that possesses them. To facilitate this access, asset owners export assets via an API that both defines and controls what operations can be performed on each asset, by whom, and under what conditions.

APIs also decouple the implementation of this access functionality from the technologies that are used to manage and store the assets. That is to say, while the assets may remain the same, the technologies used to serve and implement them can change, particularly as technological advances reduce implementation costs. APIs must preserve user access to the assets when this occurs. Thus, the lifecycle of the API follows the lifecycle of its assets and *not* the lifecycle of the surrounding technologies, which typically change at a more rapid pace.

Finally, APIs in the modern digital economy must provide standardized network-facing access so that the widest possible variety of applications and devices can access their digital assets. They must also support availability guarantees and fault management strategies associated with the assets and the implementing technologies. It is the combination of standardized, continuously available, networked access that enables a digitally based business to scale.

Thus, APIs provide three functions that are critical for the management of digital assets and artifacts. Namely, they:

1. **Implement control** over the assets, both in terms of operations and access control
2. **Protect the asset lifecycle** from technological changes driven by economics
3. **Enable scale** through standardized, networked connectivity and fault management

Because of these functions, the implementation and management of APIs can be more important than either the digital assets or the technologies that underlie them. For example, consider a company that specializes in website analytics. A change from a NoSQL database to an object store as the implementing technology should be possible without disrupting the business. Thus, the API for the analytics must remain stable while the technologies change. Similarly, the analytics data itself may be changing from day to day. The API for accessing the current data must remain constant, stable, and functional, though, or business will be interrupted.

Despite the primacy of APIs in the new digital economy, however, little technology has yet been developed to implement *API governance* — combined policy, implementation, and deployment control — in a commercial context. Good technologies exist for managing digital assets and for developing both hardware and software necessary to implement digital assets (including the necessary APIs). A few technologies[2, 3] are emerging for packaging and cataloging APIs. Yet technologies for providing stewardship of APIs through all phases of governance are rare.

INTRODUCING AppScale

In this article, we describe a strategy for implementing API governance using AppScale, a distributed software platform for managing, unifying, delivering, and composing APIs in a commercial setting. AppScale implements a set of core services that are specifically designed to implement high-level APIs in a consistent, unified

way. Using such a platform to implement APIs for commercial digital assets offers several advantages with respect to API governance. In addition to the typical API management features (cataloging, search, deployment support, etc.), AppScale focuses on the following capabilities:

- **Change control.** When API changes are necessary, AppScale restricts how they are implemented so as to control the impact of change on API consumers. If changes need to be rolled back, AppScale returns to previous functionality consistently and completely. It enables this via API usage tracking, versioning, and compatibility checking and enforcement.
- **Consistent policy implementation.** Policies governing the use of digital assets and/or their APIs are implemented consistently across the platform regardless of the constituent technologies that are used to implement the assets themselves. Administrators specify asset properties via a single portal for access control, service levels, lifecycle, backup, and failover, which the platform applies consistently across all assets.
- **Implementation portability.** API implementation is decoupled from the implementation of the digital assets. As technologies evolve or, more problematically, devolve when they sunset, AppScale maintains API integrity by providing an intermediate abstraction layer that allows the implementations to change without impacting API consumers.
- **Monitoring and auditing.** As a platform, AppScale provides a unified fabric for monitoring and auditing API activity. By doing so, AppScale allows enterprises to gather and analyze data in the same way from digital assets that use different implementation strategies and technologies.

AppScale provides these capabilities as part of a freely available and extensible distributed open source platform. As such, AppScale can be used by enterprises for API governance and application deployment without vendor lock-in. We next describe API governance in greater detail and discuss how the AppScale design facilitates such use.

UNIFYING API GOVERNANCE

Increasingly, enterprise applications are taking the form of network-accessible services that export well-defined and access-controlled interfaces. As a result, the development process includes:

- **API development** — the process of designing and coding the software components responsible for implementing the interface
- **Service development** — the process of implementing the application logic
- **Deployment configuration** — the process (often coded as scripts) of coordinating the initiation of all application components when the application is run

Thus the term "application" in this context refers to three separate but interrelated sets of programs that implement the API, service, and deployment.

This decomposition allows the service implementation and deployment components to change w*hile the API remains the same*. In this way, application users maintain consistent, unchanging access to digital assets while the service implementations and underlying infrastructure evolve in response to advances in technology.

As a result of this modularity, the lifecycle for APIs is significantly longer than that of service or deployment implementations. Moreover, from a user perspective, APIs implement policy. Access controls, SLA specification and/or negotiation, fault and error response, and so forth are all presented to users through APIs. Changes to these policies are usually global and long-lived, making their correct implementation critical to the scalable usage of digital assets.

For these reasons, in addition to standard management functions such as installation support, software patching and upgrade, and software dependency resolution, APIs require the implementation of governance — the policies and auditing functions necessary to protect the integrity of the APIs in a unified way. A unified approach to API governance is key to managing applications at scale since the applications and the digital assets they manage are likely to be developed by different entities in a large organization. Indeed, DevOps (an organizational approach that combines development and IT operations) is designed specifically to promote scalable and Agile application development by independent suborganizations. Without unified API governance, however, the scale that this new methodology engenders can lead to a proliferation of incompatible interfaces and wasted or duplicated development effort.

Using a Platform to Ensure Consistency

To ensure consistent control over the APIs in an enterprise, our approach is to build the necessary control functionality into a complete platform that spans all

resources and assets. The platform is unique in that it is designed end-to-end so that it monitors, manages, and protects all APIs under its purview in the same way, regardless of the infrastructure or digital assets involved.

Using such a platform, enterprise management is assured that policies governing APIs are implemented globally in a consistent way. This consistency of governance permits independent application development and operation by preventing the possibility that APIs will become suddenly incompatible due to changes or innovation.

To allow the technologies that implement the APIs to change as business or engineering needs warrant, AppScale plugs in multiple competitive alternatives for each service so that enterprises can compare/contrast them and choose the technologies that the local IT organization wishes to exploit.

A PLATFORM FOR UNIFIED GOVERNANCE, DEPLOYMENT, AND MANAGEMENT OF APIs

The AppScale platform[4] is a freely available, open source runtime system for Web, cloud, and mobile applications and the services they use for their implementation. AppScale implements a set of core functions that enable consistent management of the APIs that export access to these services, across the applications and digital assets it hosts. These functions include support for:

- **Plug-in integration** — a set of abstractions interposed between APIs and platform service implementations that facilitate independent and isolated service management
- **Configuration** — a service that all applications use to specify and access their respective configuration information in a consistent way
- **Deployment** — a service that invokes and decommissions APIs and service implementations under administrator control
- **Elasticity and autoscaling** — automatic resource allocation and application scaling according to an external policy, observed runtime load characteristics, and service failures
- **Auditing and monitoring** — consistent provenance for the APIs, service implementations, and digital assets managed by the platform

The AppScale platform combines these functions within a distributed system that is packaged as a virtual machine (VM) image. Platform administrators deploy AppScale via a toolset that constructs the platform as a collection of VM instances over any cluster system that supports virtualization, including public and private cloud infrastructures as well as on-premises and managed data centers. The combination of unified automated services for managing APIs separately from service implementations, the scale realized by AppScale's distributed architecture, and its portability across scalable data center technologies make it an ideal engine for implementing API governance.

Example: API Governance and Google App Engine

To illustrate how AppScale implements governance, we now describe its support for Google App Engine (GAE). In particular, AppScale exports (mirrors) the publically available APIs of GAE so that developers can deploy any GAE application either on the GAE platform over Google's resources or on the AppScale platform on-premises, without modifying their applications. To enable this, AppScale leverages plug-in integration to link each API to an open source implementation of each service. Between each API-service pair, AppScale implements a software abstraction that maps API calls to the interface of the service implementation.

To allow the technologies that implement the APIs to change as business or engineering needs warrant, AppScale plugs in multiple competitive alternatives for each service so that enterprises can compare/contrast them and choose the technologies that the local IT organization wishes to exploit, without impacting the digital assets they deliver. If, for example, an enterprise DevOps team uses the Apache Cassandra NoSQL data store, AppScale implements the GAE abstractions using Cassandra as a back end and the GAE API code as a front end. With AppScale, the applications no longer dictate the underlying technologies that must be used, allowing the IT organization to govern its infrastructure without concern for application modification. Further, if the team decides to adopt a different storage infrastructure, AppScale simply plugs in the new technology without changing the APIs the applications use to access it.

Because the API code and back-end software technologies are integrated by the distributed AppScale

platform, they can be instrumented and monitored in a uniform way. If one or more of the software modules is/are modified, AppScale can track and report on these modifications. AppScale also supports automatic deployment of these technologies so that new code is introduced in a controlled manner and can be rolled out or rolled back in a way that is both auditable and scalable.

Since AppScale itself is portable to a variety of public cloud and on-premises software environments, it is possible to run AppScale in Google Compute Engine (GCE), Amazon's AWS, and Eucalyptus.[5] GAE applications then migrate between GAE, AppScale over GCE, AppScale over AWS, and AppScale on-premises over Eucalyptus. This deployment portability using a single, consistent platform allows IT to develop a wide variety of disaster recovery and cost management policies without the need to modify the applications.

Finally, APIs do need to change from time to time. However, it is often necessary to support applications that use the "old" API as a legacy. Because AppScale runs under the control of IT or DevOps, it will run whatever version of the API the local organization requires. Thus the organization controls the lifecycle of the APIs it uses through its business logic and not the lifecycle determined by a third-party service provider.

USE CASES

We next describe two common use cases that examine key aspects of platform-based API governance using AppScale: uniform policy implementation and implementation portability. Both cases rely heavily upon monitoring and decoupling of digital asset access via APIs from the software technologies that facilitate their delivery.

- **Uniform policy implementation.** Platform administrators can use AppScale to specify a set of policies to enforce across assets. Our most common use case employs this feature to provide uniform backup of data assets and automatic failover for the services that implement them. Administrators can specify a range of properties for data assets, including how many redundant copies to store, where to store them (locally, remotely, in any number of different public or private cloud systems, etc.), and the type of consistency that should be employed across copies. For executing services, administrators identify those that are fault tolerant and specify properties such as failover target(s) (i.e., what alternative implementations to use when a failure occurs).
- **Implementation portability.** AppScale can be used to enable businesses to avoid lock-in — the overhead associated with rewriting software in order to use alternatives to constituent software components. The implementation portability of the AppScale platform precludes lock-in in two ways. First, since the platform executes on a wide variety of deployment targets (public, private, and managed clouds, clusters, and data centers), AppScale brings cross-cloud portability to applications and services that execute over it. Second, because AppScale decouples APIs and assets from the technology that facilitates their export, administrators can easily employ different alternatives — without changing the API, the application code that uses the API, or the underlying digital assets — by selecting between them during platform deployment.

AppScale significantly simplifies API governance for these two use cases by managing the complex distributed technologies that underlie important enterprise digital asset functions and features (fault tolerance and disaster recovery) and by allowing implementation technologies to change without impacting asset access (precluding lock-in).[6-8] Moreover, AppScale provides users with a uniform way of specifying, monitoring, and customizing this functionality across assets so that developers can focus on innovation and digitally based businesses can scale their digital asset offerings.

CONCLUSION

APIs have emerged as a key component of the modern digital economy. The reason for this is that they provide access to software and data in a standardized way that is easily consumed by humans and software over a network. The aspects of APIs that are critical for their successful use by enterprises include standardized access control, protection of asset lifecycles against technological changes in their implementation ecosystem, and uniform operations and management for platform-wide features such as elasticity, availability, and fault tolerance.

Advanced cloud platforms can facilitate API governance by decoupling digital assets and their APIs from the technologies used to deliver them. The abstraction layer that enables this decoupling allows the creation of software systems that implement a set of core services that can be reused across a wide range of digital assets. AppScale is one such distributed software platform for managing, unifying, delivering, and composing APIs in a commercial setting. Additional information on the open source AppScale cloud platform can be found at www.appscale.com.

ENDNOTES

[1]Armbrust, Michael, Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ian Stoica, and Matei Zaharia. "A View of Cloud Computing." *Communications of the ACM*, Vol. 53, No. 4, April 2010.

[2]Layer7 Technologies (www.layer7tech.com).

[3]Mashery (www.mashery.com).

[4]Krintz, Chandra. "The AppScale Cloud Platform: Enabling Portable, Scalable Web Application Deployment." *Internet Computing*, Vol. 17, No. 2, March-April 2013.

[5]Nurmi, Daniel, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. "The Eucalyptus Open-Source Cloud-Computing System." Paper presented to the *International Symposium on Cluster Computing and the Grid (CCGRID '09)*, Shanghai, China, May 2009.

[6]Chris Bunch, Vaibhav Arora, Navraj Chohan, Chandra Krintz, Shashank Hedge, and Ankit Srivastava. "A Pluggable Autoscaling Service for Open Cloud PAAS Systems." Paper presented to the *IEEE Fifth International Conference on Utility and Cloud Computing*, Chicago, Illinois, USA, November 2012.

[7]Chohan, Navraj, Anand Gupta, Chris Bunch, Kowshik Prakasam, and Chandra Krintz. "Hybrid Cloud Support for Large Scale Analytics and Web Processing." Paper presented to the *3rd USENIX Conference on Web Application Development (WebApps '12)*, Boston, Massachusetts, USA, June 2012.

[8]Chohan, Navraj, Anand Gupta, Chris Bunch, Sujay Sundaram, and Chandra Krintz. "North by Northwest: Infrastructure Agnostic and Datastore Agnostic Live Migration of Private Cloud Platforms." Paper presented to the *4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '12)*. Boston, Massachusetts, USA, June 2012.

Chandra Krintz is a Professor of Computer Science at the University of California, Santa Barbara (UCSB) and cofounder of AppScale Systems, Inc. She joined the UCSB faculty in 2001 after receiving her MS and PhD degrees in computer science from the University of California, San Diego (UCSD). Dr. Krintz has mentored over 60 undergraduate and graduate students, has published numerous research articles, participates in efforts to broaden participation in computing, and is the progenitor of the AppScale project. She can be reached at ckrintz@appscale.com.

Rich Wolski is a Professor of Computer Science at UCSB and cofounder of Eucalyptus Systems, Inc. Having received his MS and PhD degrees from the University of California at Davis (while a research scientist at Lawrence Livermore National Laboratory), he has also held positions at UCSD, the University of Tennessee, the San Diego Supercomputer Center, and Lawrence Berkeley National Laboratory. Dr. Wolski has led several national-scale research efforts in the area of distributed systems and is the progenitor of the Eucalyptus open source cloud project. He can be reached at rich@cs.ucsb.edu.