Standards for Hybrid Clouds

IN HYBRID CLOUD SETTINGS, APPLICATIONS REQUIRE CAREFUL PARTITIONING AS TO WHICH COMPONENTS WILL OPERATE ON WHICH SIDE OF AN ORGANIZATION'S BOUNDARY. The control and deployment settings available within the internal private part of a hybrid cloud might not match those available from a given public cloud provider, especially if that provider only supports access through specific proprietary interfaces. This column will explore emerging and established standards applicable to these settings.

In the most recent instance of this column, I gave examples of application programming interfaces (APIs) in example scenarios related to hardware infrastructure control.[1] These examples began with adding control interfaces to individual machines or small collections of devices in a home or small office setting and ranged up to consider APIs used in large-scale datacenters. The next step is to consider hybrid settings.

Hybrid cloud scenarios arise when an organization decides for cost, security, or other considerations to keep some portion of its operations physically housed on site, while making use of resources provided outside of the organization for other features. The interaction between these two settings creates the conditions for use of a hybrid cloud approach.

ALAN SILL
Texas Tech University,
alan.sill@standards-now.org

Infrastructure Boundaries in Clouds

Software that needs to operate across boundaries in hybrid cloud settings can do so more easily when the interfaces between these components operate the same way on each side of the boundary. No matter how well-designed, controllable, understandable, or easy to use the software features of a particular public cloud provider might be, it's nonetheless a distinct disadvantage if the interfaces and APIs of that provider don't match those used by internal components on the internal, private side of a hybrid cloud deployment.

Such interfaces therefore work best when they can be deployed in a standardized, well-documented, preferably machine-readable, self-documenting way, and are easily adaptable to both sides of the hybrid cloud boundary. Of course, a cloud infrastructure has components that span both hardware and software. One goal of most cloud deployments is to disassociate these settings from each other, and therefore allow deployment and scaling of software features in ways that are as independent as possible from details of the underlying hardware.

Such dissociation is never complete. Considerations such as cost, workflow, and optimal fit of software tools to their operating environments enter into choices of deployment details such as instance size, service abstraction, and virtualization paradigms such as containers versus virtual machines. Most public cloud providers offer a wide range of user-selectable choices for these deployment details.

Because of this range, there's often a greater degree of freedom for a software designer to design around, ignore, suppress, or simply specify details of the underlying hardware configuration on which their software will run in public cloud settings than in the context of private or hybrid clouds.

The reverse of this argument is also true. In private cloud settings, the physical infrastructure is more likely to be designed to match the exact needs of a particular task or business problem, whereas in public clouds, the workflow itself generally has to be adapted to make the best use of the tools and interfaces made available by the external provider.

The most obvious feature of hybrid cloud settings is therefore that control and orchestration have to work across multiple underlying infrastructure variations, necessitating the choice of interfaces that can connect these different portions of the application or system. This feature places a premium on use of tools that are designed to work across boundaries and can function in multiple settings.

Hybrid Cloud Frameworks

These factors have led to a renewed emphasis on infrastructure frameworks that can be deployed on a variety of underlying infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) provider software stacks. The technical term associated with this process is standardization. Observers in software development and business analysis arenas sometimes shy away from using this term for fear of applying it too early, but standardization is now the explicit stated goal of some of the most modern and up-to-date activities in cloud computing.

Several software frameworks that started as individual projects are now moving in this direction. Examples include Kubernetes, Docker, and Mesos, and have led to standardization efforts such as the Open Container Initiative (OCI, www.opencontainers.org), which was created to serve as an industry forum for standardization of container-oriented virtualization.

These efforts have already borne fruit. The recent version 1.0 release of the rkt application container framework (www.github.com/coreos/rkt) by CoreOS, for example, explicitly includes standards compatibility, stating that "rkt implements the appc specification, supports the Container Networking Interface specification, and can also run Docker images."

Another notable example of successful industry-based standardization is provided by the Cloud Native Computing Foundation (CNCF, www.cncf.io), which focuses on technologies that are designed to support cloud-based workflow patterns and scale well in cloud settings. A newly formed project, the Open API Initiative (OAI, www.openapis.org), which aims to extend previous efforts of the Swagger community in the area of API description and discovery, also looks promising.

The emergence of these community-based tool development and standardization efforts is the natural reaction to the complexity that results from having multiple available approaches to a given problem. Such complexity often creates conditions such as the examples I've given, in which overall simplification can be achieved by introducing a common, well-specified unifying approach. I've discussed methods to identify and target areas in which conditions are suited for standardization in previous columns.[2]

Cloud IaaS and PaaS Standards

This column has covered formal standards related to cloud IaaS and PaaS before.[3] Some of these, such as the Open Cloud Computing Interface (OCCI) from the Open Grid Forum (OGF, www.ogf.org), the Topology and Orchestration Standard for Cloud Applications (TOSCA) from the Organisation for Advancement of Structured Information Systems (OASIS, www.oasis-open.org), and the Cloud Data Management Interface (CDMI) from the Storage Networking Industry Association (SNIA, www.snia.org), have since developed strong followings with adoption in several open source software projects. Others, such as the Cloud Infrastructure Management Interface (CIMI) from the Distributed Management Task Force (DMTF, http://www.dmtf.org) continue to make progress in their formal definition and documentation.

Each of these standards aims to address a different aspect of the complexity and confusion that currently characterizes cloud IaaS and PaaS interfaces. An extremely diverse situation currently holds among cloud providers, with each having its own tooling, deployment methods, and infrastructure control software. As discussed previously, whenever the diversity of interfaces complicates rather than simplifies programming tasks, we can expect standard protocols, APIs, and related interface tooling to emerge to simplify the deployment and operation of applications, and this condition also holds in hybrid cloud settings.

Some API designers even build in deliberately designed restrictions to prevent discovery of, or direct access to, internal features. Such restrictions can limit the usefulness of even the most powerful cloud provider services, unless the deployment also provides support for well-specified client interfaces to enable remote use by other portions of an extended system. Under such conditions, we can expect interest in cross-cutting standards that work across boundaries such as those mentioned to grow, and where successful, to flourish.

Software Tools for Hybrid Workflows

The temptation of any programmer when presented with a situation in which different parts of a problem are best solved by different individual tools is to write a script, program, or workflow tool that combines the various steps in an organized way. A related condition can also be obtained when there is some reason to build choices based on factors related to economics or speed into the solution, allowing different providers to be used or different resources to be drawn on for individual steps in the solution.

In hybrid cloud settings, the characteristic that unifies the choices to be made in a given workflow is the ability to use a common interface as the point of selection among internal options. Equivalently, the output from one step of the operation should ideally be presented in a way that feeds naturally into the next step of the workflow. Such considerations lead naturally to standardization of input and output formats and to the need for tools to adapt between interfaces where cloud provider tooling isn't yet standardized and interoperable.

This isn't a new situation, but circumstances have led the interfaces used by various cloud providers to remain separate and mutually incompatible for a very long time. As a result, several libraries and, more recently, multifunctional workflow tools have emerged to bridge over the differences between the APIs of different cloud providers. Examples include libcloud (libcloud.apache.org), deltacloud (deltacloud.apache.org), fog (www.fog.io), and the Spinnaker framework (www.spinnaker.io), recently open-sourced by Google.

Libcloud is a Python module that provides a layer of abstraction covering many different cloud provider APIs, allowing programmers to write applications that work across multiple providers. Libcloud provides an application binary interface that allows cloud interfaces to be treated as drivers, rather than forcing the code to deal directly with each cloud provider's API. To use libcloud in cloud software, the developer must include the Python module directly in the application's code base rather than treat it as a remote procedure. Libcloud has progressed to a 1.0.0 preview release and remains under active development. It has also evolved to include support for block and object storage, load balancer, and domain name services.

Circumstances have led the interfaces used by various cloud providers to remain separate and mutually incompatible for a very long time.

Deltacloud was an Apache project similar in intent to libcloud, but written in a different language (Ruby) and more limited in implementation options. It included an implementation of the CIMI standard as well as Amazon Elastic Compute Cloud (EC2) and its own internal API, but otherwise didn't progress toward community adoption. The project hasn't seen significant activity in over two years, and has probably been abandoned. Although not directly related, the fog Ruby gem has emerged with similar functionality and support for a much larger range of cloud providers.

Ultimately, whether or not they're technically successful, such multiprovider "Swiss Army knife" adapter toolkits will obviously depend on being kept up to date with changes to the underlying APIs of the different cloud providers, and hence differ in intent and function from true API standards. Whether or not they can be used in a particular setting highly depends on the situation in which they're deployed, and on the willingness of the package providers and users to adapt to changes to the provider APIs.

Spinnaker differs from these other adapter toolkits in that it aims at a broader set of problems in cloud deployment that are generally referred to as continuous delivery. Instead of just trying to adapt an API call for infrastructure control or services to the APIs of different providers, Spinnaker includes many other aspects of software integration, delivery, and deployment along with cluster management in its feature set.

Although it includes a set of adapters for different cloud providers, Spinnaker also incorporates workflow concepts such as pipelines, stages, scheduling, and triggers for various steps of cloud software management. It also includes interfaces to code repositories, continuous integration engines, image management, and an internal orchestration engine to accomplish a wide variety of tasks that are important in maintaining a cloud software deployment and delivery. Each of these stages can in principle be hosted internally within an organization or on one of several supported cloud providers, making Spinnaker attractive for use in hybrid cloud settings.

As standards emerge from any of the previously discussed efforts, it would seem natural to see them included in such multifunctional tools, and eventually (if successful) to supplant and replace differences among the different cloud providers. It's easy to see some of the container standardization efforts mentioned here, for example, as candidates to be deployed within and among these tools.

AS ALWAYS, THIS DISCUSSION ONLY REPRESENTS MY OWN OPINION DERIVED FROM WORKING WITH THE PROJECTS, STANDARDS, AND SOFTWARE PRODUCTS MENTIONED. I'm interested in hearing your opinion and experience, and I'm sure other readers of the magazine would also appreciate such input.

Please respond with your opinions on this topic or on those explored in previous columns. Let us know what you think, and please also include any news you think the community should know about the general areas of cloud standards, compliance, or related topics. We're always open to article submissions. I'm happy to review ideas for such submissions, or for proposed guest columns. I can be reached for this purpose at alan.sill@standards-now.org.

References

1. A. Sill, "When to Use Standards-Based APIs (Part 2)," *IEEE Cloud Computing*, vol. 2, no. 6, 2015, pp. 80–84; doi:10.1109/MCC.2015.120.
2. A. Sill, "Socioeconomics of Cloud Standards," *IEEE Cloud Computing*, vol. 2, no. 3, 2015, pp. 8–11, doi:10.1109/MCC.2015.59.
3. A. Sill, "Cloud Standards News and Updates," *IEEE Cloud Computing*, vol. 2, no. 1, 2015, pp. 72–75, doi:10.1109/MCC.2015.4.

ALAN SILL *is interim senior director at the High Performance Computing Center at Texas Tech University, where he's also site director for the US National Science Foundation Cloud and Autonomic Computing Center and adjunct professor of physics. Sill has a PhD in particle physics from American University. He is an active member of IEEE, the Distributed Management Task Force, and the TeleManagement Forum, and he serves as president for the Open Grid Forum. For further details, visit http://cac.ttu.edu or contact him at alan.sill@standards-now.org.*