UDC 004.514

CLOUD APPLICATION ARCHITECTURES: CLOUD SERVICE MODELS

ARTYOM SHYMUKOVICH, YAUHEN SUKHAREU
Polotsk State University, Belarus

Introduction. Cloud computing has recently emerged as one of the buzzwords in the ICT industry. Numerous IT vendors are promising to offer computation, storage, and application hosting services and to provide coverage in several continents, offering service-level agreements (SLA)-backed performance and uptime promises for their services. While these "clouds" are the natural evolution of traditional data centers, they are distinguished by exposing resources (computation, data/storage, and applications) as standards-based Web services and following a "utility" pricing model where customers are charged based on their utilization of computational resources, storage, and transfer of data. They offer subscription-based access to infrastructure, platforms, and applications that are popularly referred to as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). While these emerging services have increased interoperability and usability and reduced the cost of computation, application hosting, and content storage and delivery by several orders of magnitude, there is significant complexity involved in ensuring that applications and services can scale as needed to achieve consistent and reliable operation under peak loads.

Currently, expert developers are required to implement cloud services. Cloud vendors, researchers, and practitioners alike are working to ensure that potential users are educated about the benefits of cloud computing and the best way to harness the full potential of the cloud. However, being a new and popular paradigm, the very definition of cloud computing depends on which computing expert is asked. So, while the realization of true utility computing appears closer than ever, its acceptance is currently restricted to cloud experts due to the perceived complexities of interacting with cloud computing providers.

The Cloud. The cloud is not simply the latest fashionable term for the Internet. Though the Internet is a necessary foundation for the cloud, the cloud is something more than the Internet. The cloud is where you go to use technology when you need it, for as long as you need it, and not a minute more. You do not install anything on your desktop, and you do not pay for the technology when you are not using it. The cloud can be both software and infrastructure. It can be an application you access through the Web or a server that you provision exactly when you need it. Whether a service is software or hardware, the following is a simple test to determine whether that service is a cloud service:

If you can walk into any library or Internet cafe and sit down at any computer without preference for operating system or browser and access a service, that service is cloud-based. I have defined three criteria I use in discussions on whether a particular service is a cloud service:

- The service is accessible via a web browser (nonproprietary) or web services API.
- Zero capital expenditure is necessary to get started.
- You pay only for what you use as you use it.

We should take a moment to understand a variety of cloud infrastructure models. It would be easy to contrast these services if there were fine dividing lines among them, but instead, they represent a continuum from managed services through something people call Infrastructure as a Service (IaaS) to Platform as a Service (PaaS) and Software as a Service (SaaS).

Cloud service models. Choosing the right service model is a critical success factor for delivering cloud-based solutions. In order to choose the right service model or combination of service models, one must fully understand what each service model is and what responsibilities the cloud service providers assume versus the responsibilities the cloud service consumer assumes.

There are three cloud service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Each cloud service model provides a level of abstraction that reduces the efforts required by the service consumer to build and deploy systems. In a traditional on-premises data center, the IT team has to build and manage everything. Whether the team is building proprietary solutions from scratch or purchasing commercial software products, they have to install and manage one-to-many servers, develop and install the software, ensure that the proper levels of security are applied, apply patches routinely (operating system, firmware, application, database, and so on), and much more. Each cloud service model provides levels of abstraction and automation for these tasks, thus providing more agility to the cloud service consumers so they can focus more time on their business problems and less time on managing infrastructure.

Figure 1 displays what is called the cloud stack. At the bottom is the traditional data center, which may have some virtualization in place but does not have any of the characteristics of cloud computing. The five characteristics of cloud computing are network access, elasticity, resource pooling, measured service, and on-demand self-service.

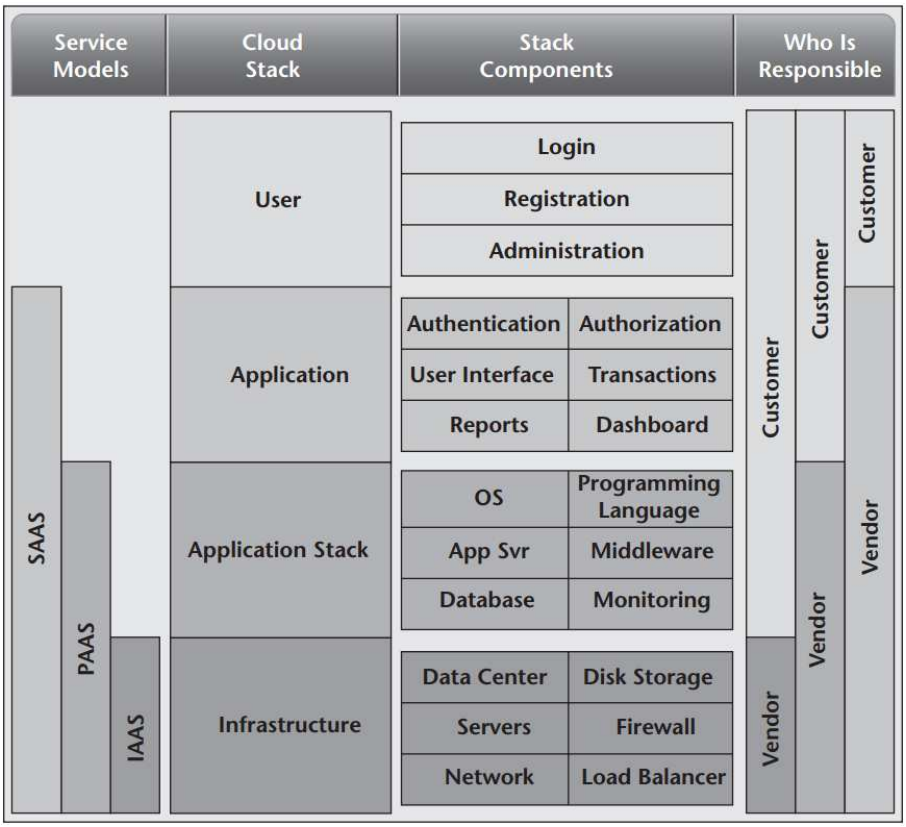

Fig 1. Cloud service models

IaaS

The National Institute of Standards and Technology (NIST) defines IaaS as: "The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications and possibly limited control of select networking components (e.g., host firewalls)."

The Cloud Security Alliance (CSA), a standards organization for cloud security, states that IaaS: "Delivers computer infrastructure (typically a platform virtualization environment) as a service, along with raw storage and networking. Rather than purchasing servers, software, data center space, or network equipment, clients instead buy those resources as a fully outsourced service."

With IaaS, many of the tasks related to managing and maintaining a physical data center and physical infrastructure (servers, disk storage, networking, and so forth) are abstracted and available as a collection of services that can be accessed and automated from code- and/or web-based management consoles. Developers still have to design and code entire applications and administrators still need to install, manage, and patch third-party solutions, but there is no physical infrastructure to manage anymore. Gone are the long procurement cycles where people would order physical hardware from vendors that would ship the hardware to the buyer who then had to unpackage, assemble, and install the hardware, which consumed space within a data center. With IaaS, the virtual infrastructure is available on demand and can be up and running in minutes by calling an application programming interface (API) or launching from a web-based management console. Like utilities such as electricity or water, virtual infrastructure is a metered service that costs money when it is powered on and in use, but stops accumulating costs when it is turned off. In summary, IaaS provides virtual data center capabilities so service consumers can focus more on building and managing applications and less on managing data centers and infrastructure. There are several IaaS vendors in the marketplace and too many to name in this book. The most mature and widely used IaaS cloud service provider is Amazon Web Services (AWS). Rackspace and GoGrid are also early pioneers in this space. OpenStack is an open source project that provides IaaS capabilities for those consumers who want to avoid vendor lock-in and want the control to build their own IaaS capabilities in-house, which is referred to as a private cloud. There are a number of companies that are building IaaS solutions on top of OpenStack similar to how there are many different distributions of Linux.

PaaS

The next level up on the stack is PaaS. What IaaS is to infrastructure, PaaS is to the applications. PaaS sits on top of IaaS and abstracts much of the standard application stack–level functions and provides those functions as a service. For example, developers designing high-scaling systems often have to write a large amount of code to handle caching, asynchronous messaging, database scaling, and much more. Many PaaS solutions provide those capabilities as a service so the developers can focus on business logic and not reinvent the wheel by coding for underlying IT "plumbing."

NIST defines PaaS as: "The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."

The CSA describes PaaS as: "The delivery of a computing platform and solution stack as a service. PaaS offerings facilitate deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities."

The CSA also mentions that PaaS services are available entirely from the Internet. PaaS vendors manage the application platform and provide the developers with a suite of tools to expedite the development process. Developers give up a degree of flexibility with PaaS because they are constrained by the tools and the software stacks that the PaaS vendor offers. The developers also have little-to-no control over lower-level software controls like memory allocation and stack configurations (examples: number of threads, amount of cache, patch levels, etc.). The PaaS vendors control all of that and may even throttle how much compute power a service consumer can use so that the vendor can ensure the platform scales equally for everyone. Chapter 5 ("Choosing the Right Cloud Service Model") explores these service model characteristics in great detail. Early PaaS pioneers like Force.com, Google Apps Engine, and Microsoft Azure dictated both the platform stack and the underlying infrastructure to developers. Force.com dictates that developers write in Apex code and the underlying infrastructure must be on Force.com 's data center. Google Apps Engine originally required that developers code in Python and on the Google data center while Azure originally required .NET technologies on Microsoft data centers. A new breed of PaaS vendors have emerged and have created an open PaaS environment where consumers can implement the PaaS platform on the infrastructure of their choice and with many options for the development stack, including PHP, Ruby, Python, Node.js, and others. This approach is critical for widespread adoption by enterprises since many enterprises require or prefer to keep some or all of the application on-premises in a private cloud. Often, large enterprises leverage hybrid clouds by keeping their data in a private cloud and moving non-mission-critical components into the public cloud. Both Google and Microsoft now support multiple development languages, whereas in the past they only supported one. Heroku and Engine Yard are examples of mature public PaaS solutions that provide multiple stacks for developers, although at the time of the writing of this book they can be deployed only on AWS. Another huge advantage of PaaS is that these platforms integrate with numerous third-party software solutions, which are often referred to as plugins, add-ons, or extensions.

By leveraging APIs to access numerous third-party solutions, developers can provide fail over, high service level agreements (SLAs), and achieve huge gains in speed to market and cost efficiency since they don't have to manage and maintain the technology behind the APIs. This is the power of PaaS, where developers can quickly assemble a collection of mature and proven third-party solutions simply by calling APIs and not having to go through a procurement process followed by an implementation process for each third-party tool. PaaS allows companies to focus on their core competencies and integrate with the best-of-breed tools in the marketplace. PaaS is the least mature of the three cloud service models but analysts predict a huge boom in the PaaS marketplace in the next several years.

SaaS

At the top of the stack is SaaS. SaaS is a complete application delivered as a service to the service consumer. The service consumer has only to configure some application-specific parameters and manage users. The service provider handles all of the infrastructure, all of the application logic, all deployments, and everything pertaining to the delivery of the product or service. Some very common SaaS applications are customer relationship management (CRM), enterprise resource planning (ERP), payroll, accounting, and other common business software. SaaS solutions are extremely common for non-core-competency functionality. Companies choose to rely on SaaS solutions for non-core functions so they do not have to support the application infrastructure, provide maintenance, and hire staff to manage it all. Instead they pay a subscription fee and simply use the service over the Internet as a browser-based service. NIST defines SaaS as:

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the

underlying cloud infrastructure, including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

REFERENCES

1. Buyya, R. Cloud Computing: Principles and Paradigms / R. Buyya, J. Broberg, A. M. Goscinski. – Hoboken : Wiley, 2011. – 664 p.
2 Reese, G. Cloud Application Architectures: Building Applications and Infrastructure in the Cloud / G. Reese. – Sebastopol : O'Reilly Media, 2009. – 208 p.
3. Kavis, M.J. Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS) / M.J. Kavis. – Hoboken : Wiley, 2014. – 224 p.