Los Muchos Significados del *"cómputo en la nube"*

DESMITIFICANDO EL CONCEPTO

Uno de los términos ampliamente en boga en nuestro campo hoy en día es el "cómputo en la nube". Tan en boga que me parece que se ha convertido en una palabra mágica, bastante hueca y sintomática de querer parecer en sintonía con los últimos desarrollos tecnológicos, sin comprenderlos en realidad. Además, al ser una frase que de golpe comenzó a escucharse con tanta insistencia, asumimos que es una estrategia nueva, una idea posibilitada por los nuevos avances tecnológicos — Y esto dista de ser el caso. En esta columna busco clarificar los conceptos y tipos básicos de cómputo en la nube, sus principales ventjas y desventajas, y brevemente encontrar paralelos con casos documentados de estos "novedosos" conceptos.

No critico, ojo, a las estrategias que están detrás de este sonoro término — Muy por el contrario, resultan muy convenientes a la hora de implementar, y como profesionales del desarrollo de software tenemos la obligación de estar familiarizados con ellas. Lo que critico es el abuso de un término ambíguo, que suele poner en problemas a los implementadores, al no estar claramente comprendido.

La mayor parte de las referencias que consulté mencionan a tres capas o niveles principales: El software –o la aplicación– como un servicio (Software as a Service, SaaS), la plataforma como un servicio (Platform as a Service, PaaS) y la infraestructura o el hardware como un servicio (Infrastructure as a Service, IaaS). Mi punto de vista, es que más que un nivel distinto, IaaS es un conjunto de cualidades adicionales que dan valor agregado a una oferta de PaaS.

Gunnar Wolf es administrador de sistemas para el Instituto de Investigaciones Económicas de la UNAM y desarrollador del proyecto Debian GNU/Linux.
www.gwolf.org

SaaS: Bloques de construcción

Cuando uno de nuestros sistemas utiliza un API público remoto, o cuando para nuestro flujo de trabajo empleamos a una aplicación que reside y es administrada fuera de nuestra organización, estamos empleando SaaS. Nuestro proveedor puede cobrar su servicio de diversas maneras: a través de una renta directa, requiriendo del despliegue de publicidad, recibiendo autorización para recolectar nuestra información y hacer minería de datos sobre de ella — Hay una enorme variedad de esquemas, que refleja la gran variedad de niveles de integración que se puede dar a los servicios.

Notarán que la principal diferencia con los esquemas tradicionales cliente-servidor con que hemos trabajado desde hace décadas es el uso explícito de un proveedor externo — Sin embargo, técnicamente, nuestros sistemas no seguirán una lógica o un proceso de desarrollo diferente de como lo vienen haciendo por décadas. Al mismo tiempo, sí hay una importante diferencia: Al no controlar nosotros el proceso que ocurre dentro de uno de nuestros subsistemas, nos vemos forzados a hacer bien lo que ya deberíamos estar haciendo: Implementar una separación limpia de responsabilidades entre los componentes, utilizando exclusivamente las interfaces publicadas explícitamente — No brincarnos las capas de abstracción, como muchas veces estaríamos tentados a hacerlo en un desarrollo interno.

Hay algo que no podemos perder de vista cuando empleamos un servicio de aplicaciones como parte de nuestra infraestructura en sistemas: Al depender de un tercero, ponemos en sus manos parte importante de nuestras promesas (y, por tanto, requerimientos), en tanto a disponibilidad, protección de datos y continuidad de la operación. Respecto a la disponibilidad, resulta natural que una falla en el servicio de cualquiera de nuestros proveedores de aplicaciones llevará a que nuestro sistema presente información incompleta o errónea a sus usuarios. En cuanto a la proteción de datos, no sólo debemos tener en cuenta las implicaciones directas (el proveedor tendrá acceso a todos los datos específicos que le enviemos para operar), sino también las indirectas: todo ataque por medio del cual un agente hostil pueda llevar a cabo recolección de información sobre de nuestros proveedores resultará en la probable divulgación de información interna nuestra, e incluso el mismo proveedor puede estar realizando un seguimiento estadístico de nuestros patrones de uso, revelando mucho más de lo que podríamos querer darle. En tanto a la continuidad de operación, si el proveedor del servicio decide cambiar los términos bajo los cuales brinda sus recursos, o si sencillamente deja de operar, podemos quedar sin una alternativa disponible para parte importante de nuestro flujo de trabajo. Como corolario de la dependencia de terceros, al depender de aplicaciones como servicio perdemos la capacidad de planear las actualizaciones de nuestra infraestructura — Al depender de proveedores externos, en el momento en que cualquiera de sus APIs cambia, nos vemos forzados a actualizar de inmediato. Nuestros servicios en producción pueden fallar, y el mero concepto de "rama estable" pierde buena parte de su atractivo [1].

"Lo que critico es el abuso de un término ambíguo"

PaaS+IaaS: Flexibilidad en el uso de recursos

Mientras que en el apartado anterior partíamos de que –desde el punto de vista del usuario– hay una aplicación central que corre en sus instalaciones y consume procesamiento realizado por componentes distribuídos por el mundo ("en la nube"), aquí la aplicación completa será desplegada en el proveedor de servicios. El usuario deja de emplear un servidor –o una granja de servidores– para consumir sólo los recursos. Y, consecuentemente, el esquema de utilidad para un proveedor de servicios bajo esta modalidad puede ir desde la renta a costo fijo hasta un cobro por uso de recursos, típicamente permitiendo reconfiguraciones dinámicas del paquete (del conjunto máximo de recursos) como respuesta a la demanda del sistema.

La justificación detrás de este concepto es que los servidores "en casa" representan un gasto demasiado grande — Compra y actualización periódica de equipo, consumo eléctrico, uso de espacio físico y de ancho de banda, siendo que las instalaciones de la mayor parte de los usuarios no cuentan siquiera con un centro de datos. Además, todo servidor debe ser adquirido contemplando la carga estimada que sostendrá, pero contemplando un espacio para sobreconsumo en picos de actividad no planificable. En cambio, un proveedor masivo de servicios balancea el exceso de demanda de un usuario con la reducción de demanda de otro, permitiendo un menor sobredimensionamiento — Y una reducción neta de precios.

Nuevamente, apreciarán que la idea no es nueva — Los proveedores de presencia o de hosting existen desde hace muchos años en Internet. Más allá aún, precisamente este esquema fue planteado a mediados de los 1960, cuando fue desarrollado el sistema MULTICS [2]. La principal diferencia con el esquema tradicional de renta de servidores es que bajo este esquema, el usuario deja de ser responsable de la administración incluso del servidor en el que se ejecutarán sus aplicaciones; el proveedor de servicios ofrece una cartera de plataformas administradas. Las plataformas pueden ser desde aplicaciones medianamente complejas, como plataformas Web que llevan un grado no trivial de configuración y representan una solución completa e integrada, hasta marcos para el desarrollo de sistemas (como Rails, Struts, Django, GWT), para los cuales los clientes sólo proveen el código específico de su aplicación, y aprovechan todo el andamiaje común que ya existe para una amplia base de clientes.

Respecto a consideraciones de seguridad y confiabilidad: Si bien al emplear PaaS no caemos ya en la trampa descrita en el primer apartado respecto a emplear código del cual no tenemos más conocimiento que el API, éste esquema sigue depositando todos nuestros datos en control del proveedor, en infraestructura compartida, con lo cual la probabilidad de que un ataque a un sistema sin relación aparente con el nuestro puede dar a un intruso acceso pleno a nuestra información. El riesgo de que el proveedor cambie sus políticas de cobro a un esquema que no nos convenga se reduce también fuertemente, dado que en principio tenemos todo lo necesario para replicar la instalación con cualquier otro proveedor, o incluso migrar a una solución "en casa". Sin embargo, seguiremos sujetos a los "días bandera" de actualización forzada, dado que el proveedor típicamente ofrecerá la misma versión base a todos sus clientes — Y este factor puede dificultarnos una migración entre proveedores. Hablando meramente de la conveniencia: Podemos enviar a uno de nuestros sistemas a la nube dados sus requisitos de almacenamiento de información, o de ancho de banda. Ahora bien, ¿cómo estamos realizando nuestros respaldos? Para poder hacer frente a contingencias, contar con una estrategia de respaldos buena y probada es mucho más importante aún que cuando hablamos de despliegues en infraestructura propia.

Por último, respecto al IaaS: Este resulta el más nebuloso y ambiguo — Ya no se trata de sistemas o entornos para que éstos se ejecuten, sino que de recursos de cómputo que están a nuestra disposición: Espacio en disco o en memoria, "ticks" de procesador, ajuste de ancho de banda, como componentes discretos y ajustables en vivo — Atributos que se apoyan fuertemente en la virtualización para poder ser desplegados y controlados. Hay quien incluye la virtualización de escritorios dentro de la definición de IaaS, pero si mucho, eso debe ser visto como consolidación de recursos y mejoría en el aprovechamiento de recursos, pero por calidad en el servicio debe hacerse utilizando recursos internos a la organización — Y por tanto, se aleja de las definiciones básicas de en la nube.

Llevamos años haciendo cómputo en la nube de un tipo o de otro. Al desmitificarlo un poco, espero abonar a una mejor utilización. Y al hablar acerca de los nada ignorables riesgos que conlleva, apunto a la cautela que debemos tener al adoptarlo, no entrar de lleno sólo porque es la moda. SG

››Por Gunnar Wolf

Notas:

[1] Un ejemplo de esto se dio en agosto de este año cuando Twitter cambió su esquema de autenticación, obligando a miles de aplicaciones a migrar su código al nuevo esquema.

[2] El sistema operativo MULTICS partía del planteamiento de que el cómputo se convertiría en un servicio provisto centralmente, como la electricidad o el teléfono. http://www.multicians.org