Christopher Kuzemka
Two Key Technical Concepts of Cloud Computing
Fordham University

The definition of cloud computing was intentionally introduced as vague and ambiguous at the very start of this course. The reasoning for ambiguity by the professor was meant to prop up a major aim for the students - to ultimately create a definition that best encapsulated the spirit of the field by the end of the course. Throughout its duration, several categorical classifications were provided and divided as individual subjects. Today we isolated two of them to begin an interpreted definition of cloud computing and the categories we chose to discuss are “web servicing” and “virtualization”.

A common example many consumers may associate with the term “cloud computing”, is the idea of storing personal data somewhere as a backup away from a local storage unit. As society becomes more reliant on data to accomplish goals or to document history, the importance of its storage blossoms and the market for systems in place to back up data in case of loss or other miscellaneous reasons proves itself to be valuable. As a result, cloud-based storage systems or repositories become more readily offered - examples of such include Github, Google Drive, and iCloud. Extending beyond storage systems, infrastructural services emerge when data not only must be stored but also be processed quickly and as a result, we then find businesses offering cloud enterprise services such as Microsoft Azure, Google Cloud, and Amazon Web Services (AWS). With infrastructure for data storage and data processing available, the power one has for computation grows and other feats could be achieved such as offering the capability to host structured video streaming sites (such as Youtube and Netflix), the capability to host social media sites (such as Facebook, Twitter, and Instagram), the capability to host online gaming (such as Xbox Game Pass/Cloud gaming), or the capability to create any customized platform for any type of business (such as Fordham University’s student portal).

In these examples, the user is not typically responsible for keeping any form of software installed locally on their devices and rather a heavy portion of any software’s infrastructure is hosted elsewhere - what we may refer to as a “cloud”. The commonality, beyond just data wrangling in the above web services mentioned, is the reliance of the Internet for these platforms to operate appropriately, and sometimes reliance on the World Wide Web. The major benefits for this form of cloud computing is its accessibility and portability for sophisticated software; often reliant on a client-server model running on internet protocols. Internet protocols ensure a standardized method where a client (the user) accesses such web services housed on remote servers and are bridged easily through a lightweight medium such as a web browser or other localized dedicated software. What cloud computing importantly offers here is convenience and broader streamlined provision of such service. Here, a user need not be concerned with requiring stronger physical hardware (in most cases) as much of the computational overhead is supported by a trusted provider; and this gives a lot of flexibility and control for the provider to optimally develop the best sophisticated software for the best user experience in handling a specific set of goals.

Large scale web-services are typically hosted on large scale compute clusters located in various parts of the world. This relationship is commonly required for any enterprise service to conveniently reach out to thousands of users. A layman would think that to make this possible, one would need a “very large computer” with total control over it so that developed software could work optimally. This is true for enterprise level web services; a large computer (or the perception of one) is often needed to make specific software work for a vast majority of people leveraging various different personal hardware (like cellphone browsers, gaming hardware, or totally unique operating systems). Sometimes, there simply doesn’t exist a “perfect” computer that best deploys a service and with the concept of cluster computing, various commodity-level hardware options can limit the capabilities and deployability for software. Collectivized hardware options could include differentiated sub-sectional specs among commodity level nodes or even differently manufactured nodes. Sometimes, the best computers a developer needs to make their software work would just fail and threaten the stability or integrity of any offered service.

The concept of “virtualization” addresses some of these environmental hurdles that developers face, to best offer the flexibility of service usage and ensure the best deployment of their software. In a superficial definition, it is considered to be the practice of emulating hardware through software. While this description has focus, we can claim a more broad description that virtualization is the “developer solution” to deploy any form of software in the best environment that makes the software run smoothly. There are 5 levels of virtualization that are commonly categorized: the ISA level, the hypervisor level, the container level, the library level, and the application level. An ISA level can interpret native OS architecture and transcribe it in real-time for a different architecture. A hypervisor can house a complete guest OS on top of a host machine’s native OS. A container is an isolated deployable environment primarily dedicated and optimized to run specific software without the need for massive global OS configurations. Library virtualization can redirect API calls of a native OS to work for unique software architecture that may be optimized to work on a different OS. And an application level can encapsulate specific applications to execute independently of the host system’s configurations.

In the above descriptions of levels as a solution for flexible deployment of software, they all share a commonality that befits the grand description of cloud computing - they all mimic the idea of having a specific kind of computer architecture around when it physically isn’t present. This ability to leverage any physical hardware’s specifications and OS setup [practically] allows a near-limitless capability for a user. In essence as a user, one doesn’t necessarily have to worry about needing to buy new hardware or do mass reconfigurations to their environment to accomplish a task reliant on some specific architecture, as one can basically “spin up their own unique computer” from any sophisticated Turing-complete machine.

Across everything shared above, the concept of “computing on a cloud” ultimately comes down to these fundamentals of leveraging “hardware” that isn’t physically around, but otherwise ethereal to the user. Whether this is accomplished through networking into servers via a web-servicing platform or being on the development side of things and focusing on the deployment of such ethereal environment, both avenues will fall under what we can call “cloud computing”