Title: Rethinking the Cloud
Authors: Marten Lohstroh, Chris Shaver

The proliferation of ubiquitous, low-cost, high-bandwidth access to the Internet, both for personal computers and mobile devices, has facilitated the development of an infrastructure that provides many forms of remote data storage for businesses, organizations, and individuals. This infrastructure has become know as the *Cloud*. Data that had in the past been stored on locally managed devices, such as biographical information, documents, work projects, media, email, and contacts, are now being moved into the Cloud, and often completely removed from, or merely cached in local forms of data storage. In addition, increasingly more sensor equipped devices automatically feed their data into the Cloud, typically through a web service that is provided by the manufacturer of the device or the developer of the software that runs on it.

This infrastructure provides the advantage of being able to access anything stored in the Cloud anywhere there is an available Internet connection. At the same time, it makes data less susceptible to loss or destruction, as opposed to storage on personal devices that can easily be damaged, lost, or stolen. Access to data is mediated through a wide diversity of services, such as web applications, search tools, and social networks, often with the ability to share or exchange data with others.

Despite the advantages of the Cloud, there are a number of significant drawbacks that emerged from the infrastructure being developed by different companies and institutions, often providing storage for specific forms or niches of data coupled with particular applications and services. The close integration of services with the data stored and operated on, has lead to data being clustered in the administrative domains of the respective service providers, ultimately fragmenting data from the perspective of the user. Rather than having one's personal data all in one virtual location, provided as generic data storage, data is stratified across several proprietary platforms, stored in different formats, and potentially with great redundancy. This makes it difficult for users to track and manage their data, or exchange it across platforms. Once captured in the Cloud, access to a segment of data is limited by the specifics of the provided interface, and subject to the service agreement with the provider. This imposes a barrier for interoperability with other services, but also limits the ability for users to migrate from one platform to another.

The fate of Cloud-stored data is also tied to the companies or institutions that host each fragment. A company going out of business or changing its business strategy can cause one's personal data to become suddenly less accessible, inaccessible, compromised, or even entirely lost. Issues of trust and personal security exist as well with these providers who can profile, sell, censor, or indiscriminately modify personal information. As data accumulated on the platforms of popular Cloud services, the companies providing those services have even shown to become the means to facilitate mass surveillance.

In order to address these issues, an attractive alternative to this collection of heterogeneous proprietary platforms that interweave personal data storage with the services they offer, would be an infrastructure in which personal data could be kept separately and independently from the services that operate on it. Rather than distributing fragments of one's data over various service providers and relying upon them to access this data, a single provider of generic, reliable, and secure personal data storage could store the whole of one's personal data in one virtual location on the Internet. It would then be up to the user to grant third parties access to particular resources. This infrastructural alternative factors the Cloud into a 'data-store', which provides a common access point to both personal and proprietary data, and a set of individuals and service providers whose services and applications can interact with this data-store. Sharing would no longer be limited to specific platforms, but become universal in the same way the early Internet made access to hypertext universal through the use of the Uniform Resource Locator (URL).

This factoring can even allow much of personal data to remain completely independent of service providers. A service can be delivered in the form of a client-side application that contains templates for personal information requests to be issued to the data-store. The requests themselves can be then made by the client securely, so that the personal data does not have to ever pass through the service provider. Client-side applications can then pull personal data directly and integrate it into their interfaces. If heavy back-end processing and aggregation must be done remotely to this data, this processing can still be performed remotely, potentially on the data-store itself, but the request for this processing will be initiated on the client-side rather than by the service provider.

One of the primary demands of an infrastructure that separates data and services is the development of a language that can be used to identify and query data from data-stores, as well as specify back-end processing to be done remotely on this data. The URL has served the purpose of providing a language for identifying data in the form of files. For this reason, the language of URLs takes on the structure of a typical hierarchical file system. More broadly, the Uniform Resource Identifier (URI) appends to the URL additional information such as key-value pairs encoding state and inputs that condition a file or resource being retrieved. As web applications have become more sophisticated, this language has been leveraged to more broadly codify general information requests; the hierarchy for instance, no longer necessarily refers to an actual file system, but instead often just codifies structured information. A URI can identify an existing file, or identify information that is generated in response to requesting it.

However, this language of URIs lacks the syntax and static semantics to match the sophistication of the requests it now often codifies. Requests to RESTful interfaces and other kinds of web applications encode a domain specific request language loosely into the syntactic structures of the URI. Particularly in an infrastructure where many different applications must be able to request information from a common data-store, a richer declarative query language needed. As first step towards building the proposed infrastructure, we will develop such a language to replace the URI as a means of making complex requests for information. This language will allow for the construction of what will be called a Uniform Information Identifier (UII).

In some respects, this language will share similarities to SQL. Indeed, behind many web services is a database accessed with SQL queries. But unlike SQL, this new language cannot presuppose a particular structure of information such as the table of rows of columns structure of a traditional database. Instead, the language must be able to specify the structure of data in a flexible way, and make statements involving this structure. The aims of this presentation at ESWeek will be to both discuss the development of this new infrastructure and to discuss more specifically the details of this declarative language for making sophisticated data requests.