The cloud revolution of IT started with the obvious first step: to virtualize computing resources. Infrastructure-as-a-Service (IaaS) clouds expose vast amounts of resources, on demand, to anyone in the world; and for a fraction of the cost needed if the same resources had to be provisioned otherwise.

However, the vast majority of cloud users do not care about resources as much as they do about their applications and middleware (or stack), which they want to run efficiently and in a scalable manner on top of some IaaS cloud. Enter Platform-as-a-Service (PaaS), which results in complete stacks of software that can easily be executed and be managed on top of an IaaS. By converting the development and operations (DevOps) actions and software stack configurations into data and code, one can easily “execute” a stack description and realize it using IaaS resources.

IaaS and PaaS work together to make the cloud revolution a reality today. However, the uptake of these two groups of technologies, in order to go mainstream, face various hurdles. For instance, how do we make sure the PaaS applications work in hybrid (private and public) clouds? How do we deal with IaaS failures at the PaaS layer? How to cope with catastrophic failures? And how to allow idiosyncratic platforms to run on PaaS that generally tend to be designed for the mainstream stacks? How to take advantage of new containerization technologies, like Docker?

Providers of SaaS and solutions develop composite applications that include multiple types of component applications – web applications, analytics components, highly customizable components that require a full stack specification, etc. The component applications may depend upon one or more services. Those providers are still struggling with the deployment and management of their component applications without having a simple way of deploying and managing their solution as a whole.

For example, take Cloud Foundry, which, as we know is an industry leading Open PaaS that provides a choice of clouds, frameworks and application services. As an open source project, there is a broad community both contributing to and supporting Cloud Foundry. Cloud Foundry applications can be easily deployed via a manifest file that includes details such as how many instances to create, how much memory to allocate and what services the application needs. While this works extremely well for traditional web applications that bind to one or more services, a large number of SaaS applications are much more complex in nature. They are composite applications made up of several component applications that need to know one another. Moreover, the component applications are not all traditional Cloud Foundry applications; rather several of them depart from the application design guidelines laid out by Cloud Foundry community – for instance, certain applications require custom software stacks, some require software stacks to be collocated, others require reading and writing from local file systems, yet others require persistence or replication of session data. With Cloud Foundry, as with PaaS in general, the infrastructure becomes a black box. This arrangement is very limiting for many applications that require some amount of access to and control of the infrastructure.

Should we break the separation of IaaS and PaaS and rethink cloud architecture around a set of APIs rather than a layered stack? This would increase the flexibility, and perhaps the appeal, of PaaS. Most PaaS offerings enable multi-cloud flexibility so that there is sufficient choice in terms of public, private and hybrid clouds for application deployment. But while they support multi-cloud flexibility, they do not address other dimensions of flexibility, such as the ability to control
the infrastructure that the applications run on or the ability to deploy and manage legacy/custom applications that do not follow the prescribed best practices for PaaS.

Should we think “federated” PaaS? For example, federate the Cloud Foundry ‘cloud’ environment where traditional component applications may run, with other types of ‘clouds’ – for instance, an analytics cloud for analytics application components, and a Docker cloud for containerized application components. In our view, highly custom component applications for a solution may be defined, declared and deployed using Docker images. While traditional applications may run on Cloud Foundry, other component applications that do not fit that mold may run on one of the other clouds. We will then need an approach that would allow solution providers to treat their solutions as single logical units of deployment and management relying on mechanisms by which solutions would be described, configured, redeployed, reconfigured and versioned as a whole.

Those issues and questions are in our view critical to understand the future of PaaS. Let us explore the current avenues, dead ends, and boulevard that make the future of PaaS (and IaaS).