JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2010**

Vol. 9, No. 3, May-June 2010

On "Cloud Nine" Through Architecture

Mahesh H. Dodani, IBM, U.S.A.

1 "ARCHITECTED" CLOUD SOLUTIONS

"That's likely because "cloud computing" isn't a "thing", it's an architecture; it's an operational model, a deployment model, even a financial model, but it's not a tangible "thing" with a specific "secret ingredient" that makes it work. Cloudness is, in fact, very much like DNA: it's the way in which the individual strands of genetic material (infrastructure) are intertwined (processes) and the result of that combination that make something a cat – or a cloud - not the existence of the individual components.

So like Google and Microsoft and Salesforce and a host of other cloud computing providers across the "aaS" spectrum, they all have the same ingredients – they've just architected them in different ways to make what we call "cloud computing." Their secret is ultimately in the operational integration of servers, storage, network, and compute resources smothered in a secret sauce called "orchestration" that gives it cloudness: a dynamic infrastructure." – If a Cat has Catness Does a Cloud have Cloudness?, Lori MacVittie Blog

This paper continues with the practice of designing cloud solutions based on an architecture that I described in my first article for 2010 and followed with showing how to use the architecture to design cloud solutions in my second article. This paper focuses on a deep dive into one of the solutions that I showcased that has been implemented using the cloud architecture.

Let us start by summarizing the components and capabilities of the cloud architecture shown in Figure 1. The three main roles in this architectural model are the service consumer (left hand side), the service provider (middle) and the service creator (right hand side). The service provider hosts services which are created by the service creator, based on a management platform consisting of an Operational Support System (OSS) and a Business Support System (BSS).

Business requirements drive the cloud service offerings and the business support systems. The architecture must be able to support a range of service offerings, including infrastructure, platform and software/applications that are needed to support the business needs. These services offerings should be able to address both enterprises using cloud computing to supplement traditional IT as well as service providers that support multiple customers. Furthermore, with different cloud-suitable services emerging, the cloud

architecture will need to provide support for workload focused offerings, including analytics, application development/test, and collaboration/e-mail services along with industry specific services. The business support systems focus on managing the business side of delivering cloud services, including managing customers, accounts, orders, subscribers, etc. Underlying these management services is the need for reporting (on usage, meeting SLAs, licenses, etc.) as well as all the capabilities for charging (including billing, invoices, settlement, etc.)

Technical requirements drive the underlying IT management patterns, including a focus on handling the top adoption factors influencing cloud services – i.e. trust, security, availability, and SLA management. The main capabilities are shown within the architecture in the operational support systems. The architecture must focus on handling the major concerns of enterprises by facilitating internal/external cloud interoperability. This requires the architecture, for example, to handle licensing and security issues to span traditional IT, private and public clouds. Additionally, the architecture must support a self service paradigm to manage clouds using a portal which requires a robust and easy to use service management solution. A portal is facilitates access to the catalog of services and to manage security services. Of course, all of these services must be provided on top of a virtualized infrastructure of the underlying IT resources that are needed to provide cloud services.

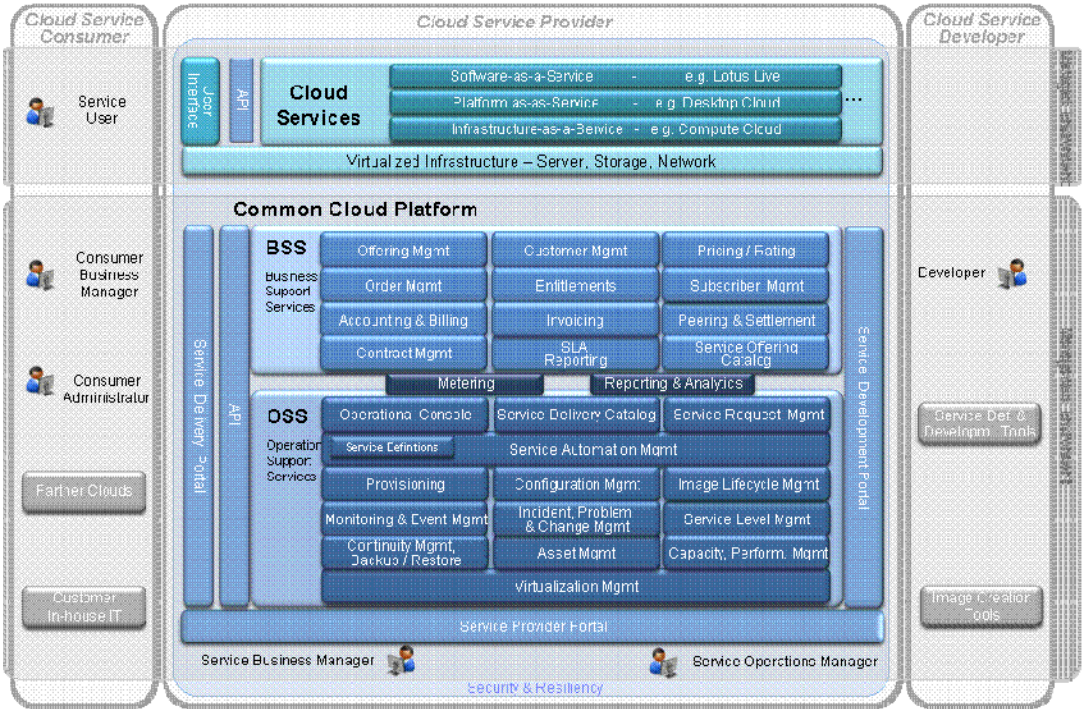

Figure 1: Cloud Reference Architecture

2 THE DEVELOPMENT AND TEST CLOUD: OBJECTIVES

The IBM Tivoli Development Services (TDS) organization provides IT services for Tivoli and other organizations in IBM Software Group in the form of lab services, host services, build/390 services, and virtual resource services to enable the software development cycle. The TDS organization faced several key business challenges. First, the organization had 24 development labs spread across the globe with minimal virtualization, resulting in much higher capital expense, management and administration costs, and less than optimal efficiency because of limited ability to reuse and share IT resources and best practices. In addition, request workflows, capacity management and administration processes were mostly manual, leading to average delivery times for new resource requests of weeks to months and driving up management costs. Tivoli software was not being leveraged globally to manage the data centers. Lastly, Tivoli's physical resources were largely underutilized, with an average utilization of about 45%.

The overall objectives of the Tivoli Development and Test Cloud include the following:

- Developers/testers reserve dev/test environments from a service catalog, use and release the environments that are handled and managed from "nearby" virtualized infrastructures.
- A central site monitors the geographically dispersed cloud environments and manages the cloud environment in areas such as performance, availability, utilization, and capacity. A key objective here is to use IBM Service Management capabilities to monitor and manage the cloud environment.
- Capacity is increased by "plugging in" a virtualized infrastructure anywhere in the world.

3 THE DEVELOPMENT AND TEST CLOUD: REQUIREMENTS

The specific functional requirements are summarized from the perspective of the primary users (developers and testers) and the administrators who monitor and manage the cloud environment. In particular, as shown in Figure 2, the developer and tester use cases facilitate interactions with the dev/test environments through the entire lifecycle, including the ability to manage dev/test environments, request these environments, use them, snapshot/restore the environments, and release them. In addition, a dev/test manager has the ability to reserve resources to cover all the development and test for the projects under their control.

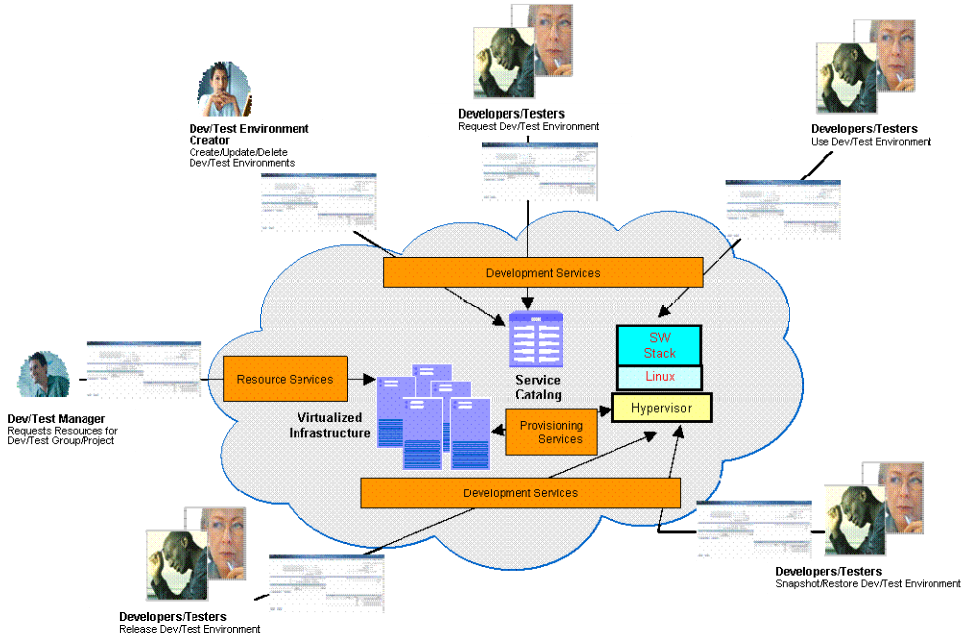

Figure 2: The Developer/Tester Use Cases

The administrator requirements, shown in Figure 3, define how the cloud environment is monitored and managed from the perspective of IT resources, virtualized environments, and cloud services. In particular, the resource administrator is interested in monitoring different types of IT resources involved in delivering cloud services, including compute, memory and storage; and the ability to manage different aspects of the IT resources, including utilization and capacity. The virtualization administrator focuses on monitoring VMs, and manages workloads associated with the cloud services as well as the performance (by increasing the IT resources allocated to the VM.) Finally, the cloud administrator monitors the entire cloud environment, and manages any incidents and events to ensure efficient and effective delivery of the cloud services to the established SLAs.

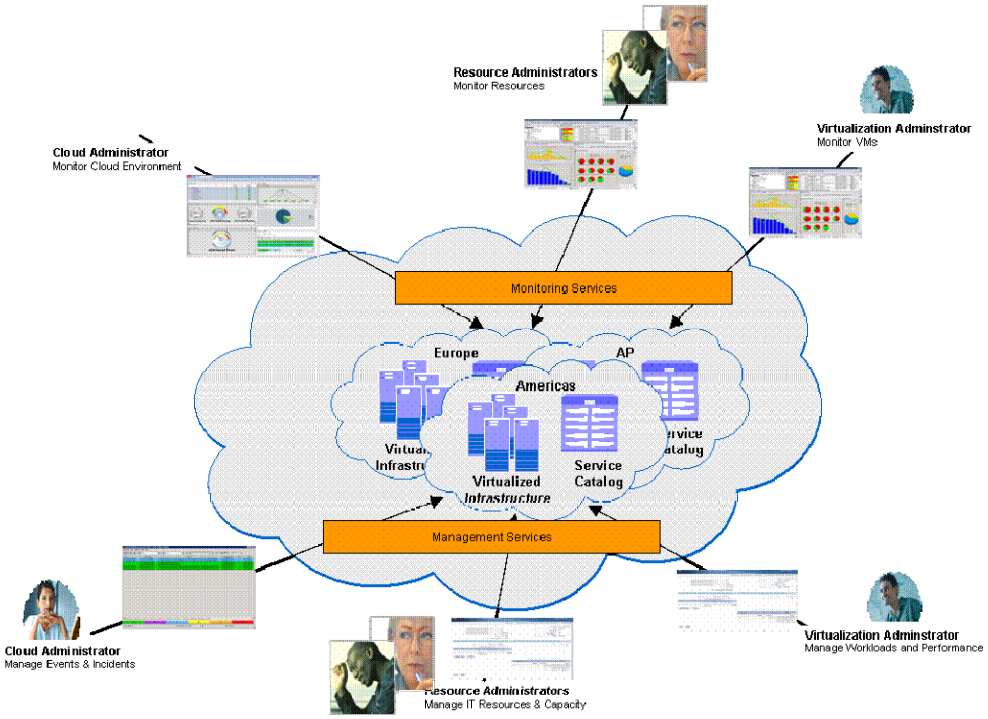

Figure 3: The Administrator Use Cases

The non-functional requirements are limited to requiring integration with the IBM enterprise LDAP to manage authentication and authorization, along with nominal requirements for availability, manageability (ease by which a new resource pool can be plugged into the cloud environment), and performance (the ability to satisfy requests for dev/test environments within a few hours.)

4 THE DEVELOPMENT AND TEST CLOUD: LOGICAL DESIGN

Based on the above requirements, the logical design of the solution is depicted in Figure 4. The intent is to have geographically distributed clouds, each capable of supporting one or more location independent resource pools. The service catalog is common across the geographic cloud, and there may be some services that are shared across geographic clouds. Each geographic cloud services its own set of users and in a later phase can make its resources usable by other geographic clouds. All the geographic clouds are monitored and managed through a central environment. Administrators can monitor and manage their cloud environments from anywhere through this central site, and make resource allocations, plan for capacities, and manage workloads across the entire Tivoli Development and Test Cloud. As shown in Figure 4 the two key components of the Tivoli Development and Test Cloud are the geographical cloud, and central monitoring and management components.

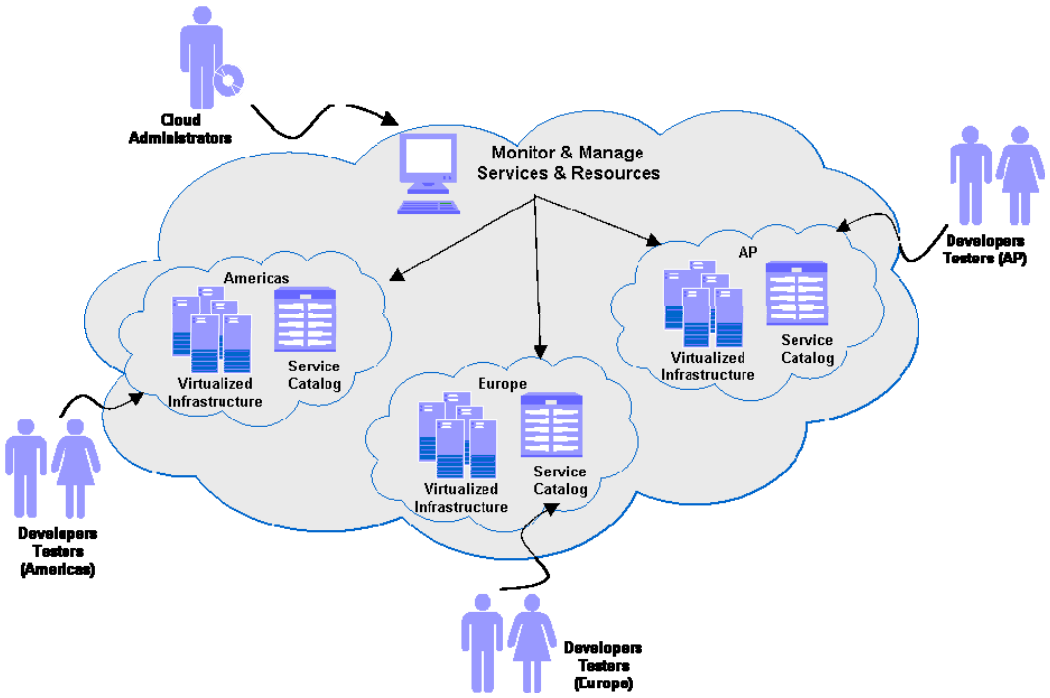

Figure 4: Logical Design of the Development and Test Cloud

The geographical cloud component includes both the managed and management environments that provide services to its users through the service catalog. The managed environment itself can be an independent resource pool available in the geography, and can be plugged in to support the needs of the Tivoli Development and Test Cloud. The service catalog provides standard services for developers and testers. Each geographic cloud can track the usage of resources based on the services requested, and provide input to support keeping track of usage against reservations of resources.

The central monitoring and management component is responsible for ensuring effective and efficient service delivery across the entire Tivoli Development and Test Cloud. It monitors all of the resources, service requests, operating systems, and energy across all the geographic cloud environments. It allows this monitoring information to be collected and presented through dashboards suitable for administrators to manage different aspects of the Tivoli Development and Test Cloud. The capabilities include the ability to monitor and manage performance and availability of the resources, the ability to monitor and manage utilization of resources and optimize it for the test workloads, and the ability to analyze the usage of resources to forecast and plan for capacity needs in the future.

The details of each of these components in the logical view: the managed environment, the geographical cloud environment and the central monitoring & management site are described in the following section. Note that the design of both the geographical cloud component and the central monitoring & management components utilize primarily the capabilities defined in the OSS layer of the architecture.

5 THE DEVELOPMENT AND TEST CLOUD: PHYSICAL DESIGN

The physical design describes the managed environment (Tivoli vCell), then shows how this managed environment is used in the geographical cloud, and finally shows how all geographical clouds get plugged into the central monitoring & management site.

Figure 5 shows the managed environment, or vCell for System x with VMWare as the hypervisor. The intent is to build a specification for each potential configuration (e.g. System z with zVM, System p with HMC, System x with KVM) to allow an IT resource pool along with an appropriate hypervisor to be easily plugged into the cloud environment. The initial focus of the implementation has been on the vCell configuration shown in Figure 5. Note that the managed environment has all of the IT resources needed to be functional in a cloud environment – including compute, storage and networking. Note that each virtualized infrastructure can handle approximately 400-600 VMs.

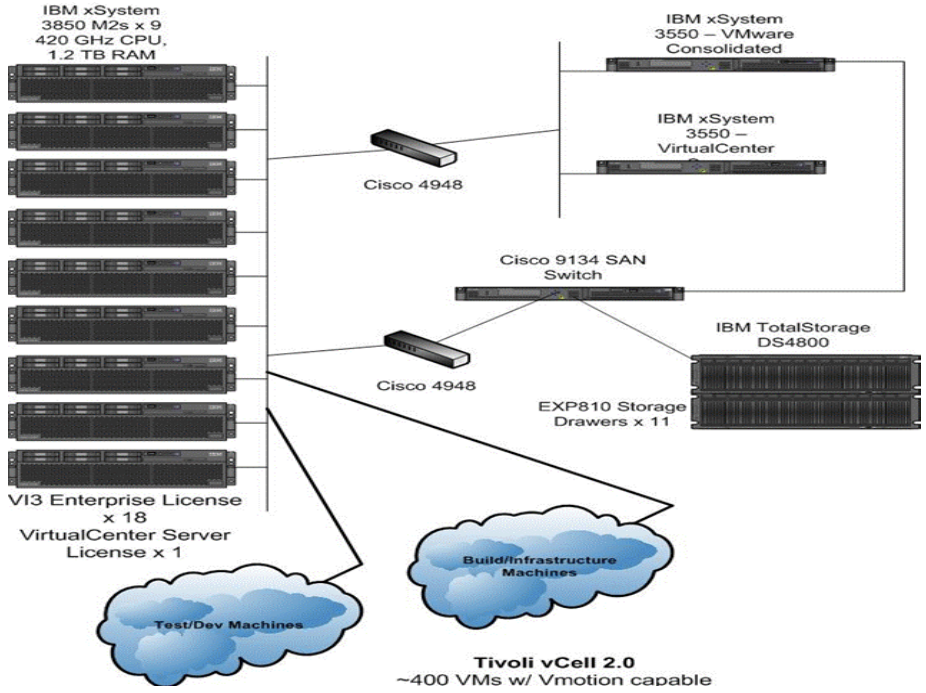

Figure 5: The Managed Environment

Each geographical cloud environment has its own management environment that delivers cloud services to its set of users as shown in Figure 6. In particular, Tivoli Service

Automation Manager manages the delivery of cloud services (dev/test environments) to the end user. Users are authenticated against the IBM LDAP (BluePages) to ease the integration of existing Tivoli developers/testers into the Tivoli Development and Test Cloud. Tivoli Service Automation Manager takes care of the entire lifecycle of delivering cloud services, including the ability to manage dev/test environments, request these environments, use them, snapshot/restore the environments, and release them. The integration with Tivoli Usage and Accounting Manager provides the additional capability of tracking the usage of the IT resources for the VMs allocated to users, and providing reports based on the usage to the dev/test managers to allow them to adjust the reservations of resources needed to support their projects.

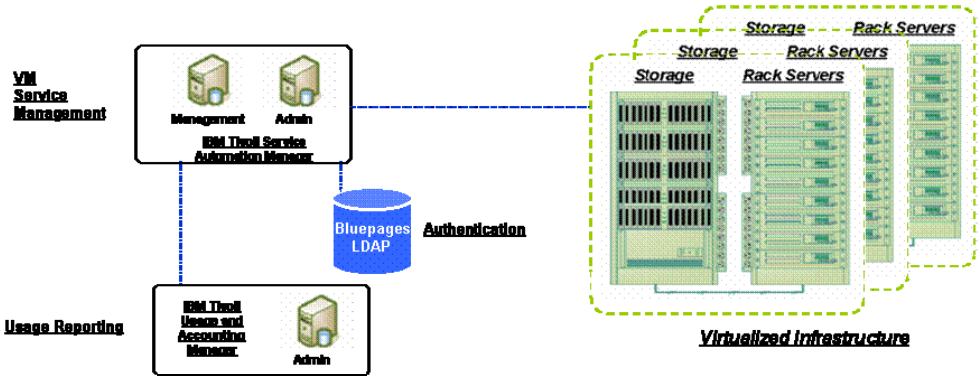

Figure 6: Geographical Cloud Environment

Figure 7 shows the central monitoring and management site design. The managed environments, VMs, and cloud services in each geographical cloud environment are monitored using different tools (both agents and agent-less monitors), and fed into the appropriate monitoring and management environments that need the information to provide support services. These monitoring results are presented on various dashboards to ease the effort of monitoring the entire Tivoli Development Cloud, and manage it appropriately. All monitoring data and events are stored in the Tivoli Data Warehouse, which then allows analysis of the data to support capacity planning and forecasting, as well as the ability to generate different reports around usage, availability and performance.

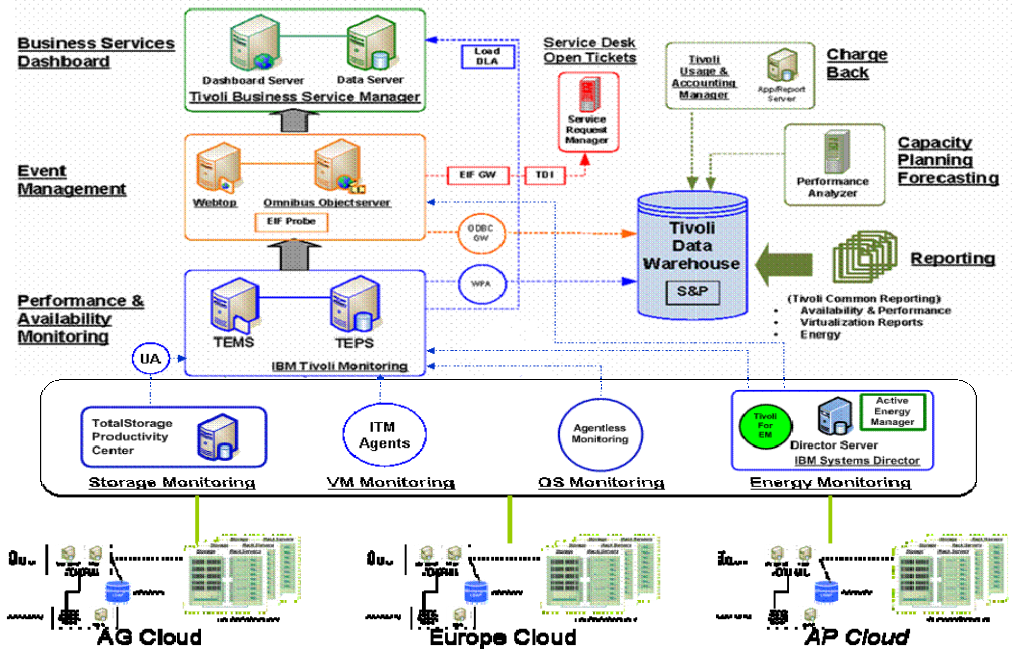

Figure 7: The Central Monitoring and Management Implementation

6 SUMMARY

This paper has described the design of a development and test cloud based on an established architectural model. Using the architecture, the Tivoli Development and Test Cloud delivered the following innovative features:

- The ability to monitor and manage geographically dispersed cloud environments through a single, central site.
- The "plug-and-play" managed environments, which allows a full managed environment to participate in supporting cloud service delivery as long as they conform to the given specification.

About the author

Mahesh Dodani is a software architect at IBM focusing on Cloud Computing. His primary interests are in enabling communities of practitioners to design and build solutions that address complex business needs and deliver value. He can be reached at dodani@us.ibm.com.