Introduction and Architecture Overview

IBM Cloud Computing Reference Architecture 2.0

Authors:
Michael Behrendt
Bernard Glasner
Petra Kopp
Robert Dieckmann
Gerd Breiter
Stefan Pappe
Heather Kreger
Ali Arsanjani
Document History

Document Location
This is a snapshot of an on-line document. Paper copies are valid only on the day they are printed. Refer to the author if you are in any doubt about the currency of this document.

Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision Date</th>
<th>Summary of Changes</th>
<th>Changes marked</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>02-28-11</td>
<td>1.0 Submission to Open Group</td>
<td>N</td>
</tr>
</tbody>
</table>

Date of this revision: 02-28-11
Date of next revision: to be defined
Contents

Document History .......................................................................................................................... 2
Document Location .......................................................................................................................... 2
Revision History .............................................................................................................................. 2

1. Introduction ............................................................................................................................... 5
   1.1. Description .......................................................................................................................... 5
   1.2. Purpose ................................................................................................................................ 5
   1.3. How to use this work product? ......................................................................................... 6
   1.4. SOA and Cloud .................................................................................................................. 6
   1.5. Using the SOA RA with the CCRA ................................................................................... 8

2. IBM Cloud Computing Reference Architecture (CC RA) Overview ...................................... 11
   2.1. Introduction ....................................................................................................................... 11
   2.2. Roles .................................................................................................................................... 12
       2.2.1. Cloud Service Consumer ......................................................................................... 12
       2.2.1.2. Cloud Service Provider ....................................................................................... 12
       2.2.1.3. Cloud Service Creator ......................................................................................... 12
   2.3. Architectural Elements ....................................................................................................... 13
       2.3.1. Cloud Service Consumer ......................................................................................... 13
       2.3.1.1. Cloud Service Integration Tools ........................................................................ 13
       2.3.1.2. Consumer In-house IT ...................................................................................... 14
       2.3.2. Cloud Service Provider .............................................................................................. 15
       2.3.2.1. Cloud Services ..................................................................................................... 15
       2.3.2.1.1. Cloud service models ...................................................................................... 16
       2.3.2.1.1.1. Infrastructure-as-a-Service ........................................................................... 17
       2.3.2.1.1.2. Platform-as-a-Service ................................................................................ 17
       2.3.2.1.1.3. Software-as-a-Service ................................................................................ 17
       2.3.2.1.1.4. Business-Process-as-a-Service ................................................................. 17
       2.3.2.1.2. Cloud service creation & ecosystem aspects .................................................. 17
       2.3.2.2. Infrastructure .......................................................................................................... 19
       2.3.2.3. Common Cloud Management Platform (CCMP) ..................................................... 21
       2.3.2.3.1. BSS – Business Support Services ................................................................... 24
       2.3.2.3.2. OSS – Operational Support Services ........................................................... 26
       2.3.2.4. Security, Resiliency, Performance & Consumability ........................................... 27
2.3.3. Cloud Service Creator ........................................................................................................28
2.3.3.1. Service Development Tools .........................................................................................28

3. CC Reference Architecture: Architectural Principles and Related Guidance ......29

3.1. CC RA-Specific Architectural Principles ...........................................................................29
3.1.1. Origins of the Term Architectural Principle .......................................................................29
3.1.2. Principle 1: Design for Cloud-Scale Efficiencies ............................................................30
3.1.3. Principle 2: Support Lean Service Management ..............................................................31
3.1.4. Principle 3: Identify and Leverage Commonalities ..........................................................33
3.1.5. Principle 4: Define and Manage Cloud Services generically along their Lifecycle ..........34

4. References ................................................................................................................................36
1. Introduction

This document serves as the definition of the IBM Cloud Computing Reference Architecture (CC RA).

A Reference Architecture (RA) provides a blueprint of a to-be-model with a well-defined scope, requirements it satisfies, and architectural decisions it realizes. By delivering best practices in a standardized, methodical way, an RA ensures consistency and quality across development and delivery projects.

The mission of the CC RA is defined as follows:

Definition of a single Cloud Computing Reference Architecture, enabling cloud-scale economics in delivering cloud services while optimizing resource and labor utilization and delivering a design blueprint for

- Cloud services, which are offered to customers
- Private, public or hybrid cloud projects
- Workload-optimized systems
- Enabling the management of multiple cloud services (across I/P/S/BPaaS) based on the same, common management platform for enabling economies of scale.

1.1. Description

The CC RA is based on real-world input from many cloud implementations across IBM. The Architecture Overview Diagram (AOD) provides overview of the fundamental architectural building blocks making up the CC RA. It also defines architectural principles serving as a guideline for creating any cloud environment.

1.2. Purpose

The Cloud Computing Reference Architecture is intended to be used as a blueprint / guide for architecting cloud implementations, driven by functional and non-functional requirements of the respective cloud implementation. Consequently, the CC RA should not be viewed as fine-granular deployment specification of just a single specific cloud implementation (and its management platform)

This document serves the following purposes:

1. This document defines the basic architectural elements and relationships which make up the IBM Cloud Computing Reference Architecture.
2. This document defines the basic architectural principles which are fundamental for delivering & managing cloud services.

The audience of the CC Reference Architecture is:

- Development teams implementing cloud services exploiting CCMP capabilities
• Development teams implementing the CCMP delivery & management capabilities for cloud services
• Practitioners implementing private clouds for customers

1.3. How to use this work product?

The architecture overview is intended to provide a common, coherent architectural structure which should be used as a basis for any cloud computing project. This allows representing the architecture of any cloud project in a consistent fashion. Existing “legacy” products and technologies as well as new cloud technologies can be mapped on the AOD to show integration points amongst the new cloud technologies and integration points between the cloud technologies and already existing ones.

The architectural principles define the fundamental principles which need to be followed when realizing a cloud. These principles need to be followed on all implementation stages (architecture, design, and implementation) and have implications across all work products.

1.4. SOA and Cloud

In order to understand the IBM CCRA it is important to understand the relationship between SOA and Cloud, not only at an architectural level, but also at a solution and service level.

Service oriented architecture (SOA) is defined by The Open Group (http://www.opengroup.org/soa/soa/def.htm) to be “an architectural style that supports service orientation. Service orientation is a way of thinking in terms of services and service-based development and the outcomes of services.”

According to NIST (http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc), “Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.”

The Service models are Cloud Infrastructure as a Service, Cloud Platform as a Service, and Cloud Software as a Service. The service models, and the fact that cloud computing is discussed in terms of the creation, delivery and consumption of cloud services, means cloud computing supports service orientation. Enterprises expose infrastructure, platforms and software as services as part of SOA solutions today. Certainly Software as a Service is not new and has been a popular topic for years.

The cloud Deployment models are Private, Community, Public, and Hybrid. These deployment models define the scope of the cloud architecture and solution, does the cloud exist strictly within the organization boundaries (private), across organization boundaries (public), or a combination (Hybrid). Certainly these scopes have been seen in SOA solutions before Cloud (while there was not a well known architectural model for them as there is in cloud computing), there are SOA solutions that exist strictly within an enterprise, or between businesses across enterprise boundaries (B2B). In fact one of the key values of SOA was to develop SOA solutions with services that are integrate between business partners, enabling outsourcing, simplifying integration and increasing agility, much like the Hybrid model. Cloud computing enables this paradigm by adding cloud-characteristics to the services being delivered & consumed.
The essential characteristics for Cloud Computing are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured Service. These characteristics can be found in requirements and SOA solutions in various organizations today, although these characteristics are optional for SOA and mandatory for cloud.

Usually, a single SOA solution does not have all of these characteristics simultaneously, unless it is a very mature organization leveraging SOA. For these organizations, each of these solutions had to be built in its entirety for that organization. The SOA solution and management of it must be built from scratch, and is not generally shared amongst organizations. Reuse is generally within an organization or an industry, not between organizations and entire communities. Service delivery and consumption aspects are a small part of the requirements for the SOA solution.

What is new about cloud is that instead of supporting these requirements per solution, the industry is trying to ‘standardize’ how these requirements are being met to enable cloud computing. Cloud architectures require a set of capabilities and ABBs to meet the NIST essential characteristics that are optional in SOA. In addition, these ABBs may be implemented in Cloud specific ways to handle scale, cost optimization and automation.

This discussion shows that Cloud computing architectures are Service Oriented architectures and adhere to architectural style that supports service orientation. Cloud solutions are SOA solutions.

The Open Group defines a service:

" A service:

- Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports)
- Is self-contained
- May be composed of other services
- Is a “black box” to consumers of the service “

Cloud services, according to The Open Group definition, are SOA services. However, not all SOA services are Cloud service because they require automated deployment and management as well as offering support in support of the Cloud characteristics.

On the architecture continuum (see TOGAF at http://www.opengroup.org/architecture/togaf9-doc/arch/chap39.html), Cloud architectures are more concrete than The Open Group’s SOA reference architecture (http://www.opengroup.org/projects/soa-ref-arch/uploads/40/19713/soa-ra-public-050609.pdf), a domain architecture scoped to service delivery and management. Principles and architectural decisions have been premade already to enable the Cloud computing architecture to be self service, network accessible, and scalable. Architectural building blocks have already been identified for Cloud solution architects to use for operational and business support. In some cases, Cloud service providers may provide well defined, maybe even standardized, management and security support and services.

For cloud, some service identification has been done (reusable, utility services, for example, to control VMs and deploy/undeploy applications or services) and implementations of services may be available from an existing services ecosystem. The existence of this services ecosystem and concrete architecture makes SOA via cloud simpler for service consumers to adopt because the designs and implementations have been provided.
The benefit of recognizing the heritage of Cloud from SOA is that the existing experience over the last 5 years and standards already available for SOA and SOA solutions can be applied to Cloud Computing and Cloud solutions.

SOA standards in The Open Group that can be applied to Cloud include:

- The Open Service Integration Maturity Model – this model helps determine the level of service use in an organization, these levels apply to the use of cloud services. Cloud computing can be seen as the “Virtualized” and “Dynamically reconfigurable” levels.
- The SOA Ontology defines service and SOA concepts which can be used as a basis for describing cloud services, though extension Ontologies should be developed for cloud.
- The SOA Reference Architecture defines the functional and cross cutting concerns and ABBs for SOA, which also applies to Cloud. This standard has been used as a basis for the IBM CCRA.
- The SOA Governance Framework defines a governance reference model and method that applies to the development of cloud services and solution portfolio and lifecycle management. Best practices for governance of Cloud solutions will need to be developed in addition to this standard.
- Security for Cloud and SOA, a joint workgroup between SOA and Cloud Workgroups in The Open Group, defines security considerations and ABBs for both Cloud and SOA.
- SOCCI, another joint SOA and Cloud Workgroup in The Open Group defines the architecture for exposing infrastructure as a service for both SOA and Cloud solutions.

Certainly functions that were optional for SOA solutions are now required for Cloud solutions, like virtualization, security across business boundaries, and service management automation. New functions and requirements are getting in focus with cloud driving experiences from the SOA world to the next level.

### 1.5. Using the SOA RA with the CCRA

The SOA RA (http://www.opengroup.org/projects/soa-ref-arch/uploads/40/19713/soa-ra-public-050609.pdf), as being standardized by The Open Group, applies to Cloud architectures and is the underlying architecture for the IBM CCRA.

The functional concerns: operational systems, service components, services, business processes and consumer interfaces; all exist in and are relevant to functional concerns for cloud architecture’s

For the Cloud architecture, there has been special focus on the:

- **Operational Layer:** Infrastructure is part of the operational systems layer, but often highlighted in Cloud architectures because Cloud imposes new requirements on infrastructure to enable broad network access, resource pooling, rapid elasticity, virtualization and scalability.
- **Service Layer:** The common cloud service types, *aaS, are identified in the services layer. These cloud service types, like other services, use and sometimes expose assets in the Operational systems layer. For cloud services, which assets are exposed is often the focus of the service type, ie within operational systems, hardware infrastructure is exposed as IaaS, and middleware is exposed as PaaS, and business process as BPaaS.
- **Business Process:** Business processes participate in a Cloud solution much like they do in SOA solutions, they can be provided as a service (BPaaS) or be the consumer of services (whether they care cloud services or not). Additionally, business processes within a cloud provider organization need to be restructured and streamlined in novel ways to meet much faster time-to-deliver, time-to-change and cost objectives.
- **Consumer Layer:** The consumer layer is more strictly and carefully separated from the services and service provider to allow pooling and substitution of cloud services or providers.
The cross cutting concerns in the SOA RA: integration, quality of service, information and governance: are important concerns for all cloud architectures and solutions, just like they are for SOA. The fact that they are cross cutting means that each of the functional layers may have interactions with capabilities in the cross cutting layers.

For the Cloud architecture, there has been special focus on the:

- **Quality of Service (QOS) layer**: The Quality of Service cross cutting concern has additional significant requirements for Cloud for management and security in order enable NISTs on-demand self-service and measured service requirements as well as IBM’s requirements for resiliency, security, performance, automated management, operational, and business support. The management support can be represented as a Common Cloud Management Platform in the SOA RA QOS layer, which includes support for operational and business support services, aka OSS and BSS. This is critical for driving economies-of-scale by delivering many cloud services based on the same foundation.

- **Governance for cloud solutions will also have some unique patterns of requirements needed to support governance across organizational boundaries.**

For the cloud ecosystem, they cloud service consumers, providers and creators are the common high level roles identified in the cloud architectures.

It is important to look at Cloud in the context of SOA, and Cloud solutions in the context of the larger SOA solutions underpinning them. This diagram shows the QOS layer details that are essential to understand for Cloud, as well as the *aaS and Infrastructure layers.
The cross cutting concerns for integration and information are still important and must be considered in the development of any Cloud architecture and solution architecture. However, cloud does not introduce any new principles or concerns to these cross cutting layers.

To make it easier to focus on the Cloud concerns rather than the SOA concerns, we can lift the Cloud concerns into its own diagram, as we show in the remainder of this document.

The concepts and architectural elements not depicted in the CCRA are still implied and present via its SOA RA heritage.

The cross cutting concerns for integration and information are still important and must be considered in the development of any Cloud architecture and solution architecture. However, cloud does not introduce any new principles or concerns to these cross cutting layers.

To make it easier to focus on the Cloud concerns rather than the SOA concerns, we can lift the Cloud concerns into its own diagram, as we show in the remainder of this document.
2. IBM Cloud Computing Reference Architecture (CC RA) Overview

2.1. Introduction

The IBM Cloud Computing Reference Architecture (CC RA, see figure 2) defines the fundamental architectural elements constituting a cloud computing environment. The CC RA is structured in a modular fashion in a sense that on its highest level of abstraction, the main roles and the corresponding architectural elements are defined allowing to drill down for each of these elements as needed. Later sections of this document will describe detailed versions of this IBM Cloud Computing Reference Architecture Overview diagram, which provide a more fine-grain view of the high-level architectural elements of the overall CC RA. As a result of specifying the CC RA, a base terminology is established, which should be used as a reference for any other cloud computing related effort.

The following sections will describe each architectural element of the CC RA depicted in figure 3 in detail.
2.2. Roles

The IBM Cloud Computing Reference Architecture defines three main roles: Cloud Service Consumer, Cloud Service Provider and Cloud Service Creator. Each role can be fulfilled by a single person or can be fulfilled by a group of people or an organization. The roles defined here intend to capture the common set of roles typically encountered in any cloud computing environment. Therefore it is important to note that depending on a particular cloud computing scenario or specific cloud implementation, there may be project-specific sub-roles defined.

2.2.1.1. Cloud Service Consumer

A cloud service consumer is an organization, a human being or an IT system that consumes (i.e., requests, uses and manages, e.g. changes quotas for users, changes CPU capacity assigned to a VM, increases maximum number of seats for a web conferencing cloud service) service instances delivered by a particular cloud service. The service consumer may be billed for all (or a subset of) its interactions with cloud service and the provisioned service instance(s).

A service consumer can also be viewed as a kind of super-role representing the party consuming services. For example, in case a credit card company is using some cloud services, the company as a whole is a service consumer relative to the provider. Within the service consumer role more specific roles may exist, such as a technical role responsible for making service consumption work from a technical perspective; and there might be a business person on the consumer side who is responsible for the financial aspects. Of course, in more simplified public cloud scenarios all of these consumer-centric roles could be collapsed into a single person, but the roles still exist.

The cloud service consumer browses the service offering catalog and triggers service instantiation and management from there. There may be cases where the interaction with the service delivery catalog is tightly embedded within the actual cloud service. In particular this is pretty common for SaaS and BPaaS cloud services where application-level virtualization is implemented, e.g. LotusLive.

2.2.1.2. Cloud Service Provider

The Cloud Service Provider has the responsibility of providing cloud services to Cloud Service Consumers. A cloud service provider is defined by the ownership of a common cloud management platform (CCMP). This ownership can either be realized by truly running a CCMP by himself or consuming one as a service.

People acting in the role of a Cloud Service Provider and a Cloud Service Consumer at the same time would be a partner of another cloud service provider reselling cloud services or consuming cloud services and adding value add functionality on top, which would in turn be provided as a cloud service.

Although defined as a separate role, it would also be possible that a Cloud Service Provider has Cloud Service Creators in the same organization, i.e. it is not necessary that Cloud Service Provider and Cloud Service Creator are in separate organizations.

2.2.1.3. Cloud Service Creator

The Cloud Service Creator is responsible for creating a cloud service, which can be run by a Cloud Service Provider and by that exposed to Cloud Service Consumers. Typically, Cloud Service Creators build their cloud services by leveraging functionality which is exposed by a Cloud Service Provider. Management functionality which is commonly needed by Cloud Service Creators is defined by the CCMP architecture. A Cloud Service Creator designs, implements and maintains runtime and management artifacts specific to a cloud service.

Just like the Cloud Service Consumer and the Cloud Service Provider, the Cloud Service Creator can be an organization or a human being. For example, an ISV company developing a cloud service is a Cloud
Service Creator, whereas obviously there could be 100’s of employees within that particular Cloud Service Creator incarnation, each of them taking on a specific business or technically focused sub-role.

It is also typical that that operations staff responsible for operating a cloud service is closely integrated with the development organization developing the service (this integration is commonly referred to as “DevOps”). This is an important aspect to achieve the delivery efficiency expected from cloud services as it allows a very short feedback loop to implement changes in the cloud service which benefit the operational efficiency of the cloud service.

### 2.3. Architectural Elements

#### 2.3.1. Cloud Service Consumer

![Figure 2: IBM CC RA – Cloud Service Integration Tools & Consumer In-house IT](image)

**2.3.1.1. Cloud Service Integration Tools**

From the perspective of a Cloud Service Consumer, it is important to be able to integrate cloud services with their on-premise IT. The functionality of Cloud Service Integration Tools is specifically relevant in the context of hybrid clouds, where seamless integrated management, usage and interoperability of cloud services in integration with on-premise IT is critical.
2.3.1.2. Consumer In-house IT

Besides IT capabilities consumed as cloud services, consumers of such IT may continue to have in-house IT, which can be managed in a traditional non-cloud fashion. In case functionality of the existing in-house IT should be integrated with cloud services consumed from a cloud service provider, the aforementioned cloud service integration tools are required. Consumer in-house IT exists across all layers of the technology stack (infrastructure, middleware, applications, business processes, service management), therefore integration with cloud services can take place on each of these layers.
2.3.2. Cloud Service Provider

2.3.2.1. Cloud Services

Cloud Services can represent any type of (IT) capability which is provided by the Cloud Service Provider to Cloud Service Consumers, implementing all cloud characteristics (self-service access, network-based access, served out of a resource pool, elastic, pay-per-use). There are four categories of Cloud Services: Infrastructure, Platform, Software or Business Process Services. In contrast to traditional (IT) services, cloud services have attributes associated with cloud computing, such as a pay-per-use model, self-service usage, flexible scaling & shared of underlying IT resources.

The management functions defined as part of the CCMP architecture are responsible for delivering instances of Cloud Services of any category to Cloud Service Consumers, the ongoing management of all Cloud service instances from a provider perspective and allowing Cloud Service Consumers to manage their Cloud Service instances in a self-service fashion.

For all cloud services there is software required implementing cloud service specifics: For IaaS, there are typically hypervisors installed on the managed hardware infrastructure, for PaaS there would a multi-
tenancy enabled middleware platform, for SaaS a (multi-tenancy enabled) end-user application and for BPaaS there typically is a multi-tenancy enabled business process engine.

Depending on the nature of the respective cloud service, there are cloud service specific management interfaces exposed, which are typically used by the management services as defined in the CCMP architecture.

Each cloud service also typically exposes APIs towards the cloud service consumer for programmatic use.

Cloud Services can be built on top of each other, e.g. a software service could consume a platform or infrastructure service as its basis, a platform service could consume an infrastructure service as its foundation. However, this is not required, i.e. a software service could also directly be built on top of bare metal infrastructure, clearly inheriting all constraints associated with such an infrastructure. In general, architectural principle #3 postulates to share as much as possible across cloud services with respect to management platform and underlying infrastructure. However, it does not require just one single, fully homogeneous infrastructure – of course, this would be the ideal goal, but given different infrastructure requirements this is not always possible. For example, if a particular cloud service has very specific infrastructure needs (with respect to reliability, latency, throughput, computational performance, hardware platform, etc.) it is clearly allowed to run this cloud service on a dedicated infrastructure (e.g. HPC cloud services would generally run on a purpose-built physical infrastructure for performance and efficiency reasons, they wouldn’t run on virtualized compute cloud service).

In the context of building cloud services on top of each other, it is important to distinguish the sharing of a common OSS/BSS (aka CCMP) structure across multiple cloud services and the usage of the actual cloud service capability by another one. An example for the first aspect would be LotusLive and Compute Cloud exploiting/getting managed by the same OSS/BSS construct. An example for the latter aspect would be LotusLive (SaaS) using the compute capacity of a compute cloud service (IaaS) vs. purchasing their own servers.

In any case, each Cloud Service offered by a Cloud Service Provider is “known” to the BSS & OSS of the Common Cloud Management Platform (otherwise the cloud service wouldn’t be visible via the service catalog and ready to be consumed). A cloud service provider offers cloud services as a result of very conscious business decisions, since taking a cloud service to market must be supported by a corresponding solid business model and investments for the development and operations (software & operations staff) of the cloud service.

2.3.2.1. Cloud service models

There are four cloud service models within the context of the IBM Cloud Computing Reference Architecture – Infrastructure-, Platform, Software- and Business-Process-as-a-Service.

IaaS, PaaS and SaaS are defined according to the “NIST Definition of Cloud Computing” [7]. There is an IBM-specific definition of BPaaS since that is not defined by the NIST.

Since this is often a point of confusion it is important to note that across all cloud service models the definition is determined by the management scope covered by the provider.

For example, in IaaS “the consumer does not manage or control the underlying cloud infrastructure […]”, in PaaS “the consumer does not manage or control the underlying cloud infrastructure, network, servers, operating systems, or storage […]”, etc.. So this essentially about the tasks the operations staff of the provider takes on, it is not about the virtualization technology being used. For example, it’s possible to use hypervisor-level virtualization to realize PaaS, SaaS or BPaaS.
2.3.2.1.1.1. Infrastructure-as-a-Service
"The capability provided to the consumer is to rent processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly select networking components (e.g., firewalls, load balancers)." [7]

2.3.2.1.1.2. Platform-as-a-Service
"The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created applications using programming languages and tools supported by the provider (e.g., java, python, .Net). The consumer does not manage or control the underlying cloud infrastructure, network, servers, operating systems, or storage, but the consumer has control over the deployed applications and possibly application hosting environment configurations." [7]

2.3.2.1.1.3. Software-as-a-Service
"The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure and accessible from various client devices through a thin client interface such as a Web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure, network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings." [7]

Software-as-a-Service is also referred to as Applications-as-a-Service since SaaS is essentially about providing applications as a service (vs. software in general). This also includes content services (e.g. video-on-demand) and higher value network services (e.g. VoIP) as typically encountered in communication service provider scenarios.

2.3.2.1.1.4. Business-Process-as-a-Service
"Business process services are any business process (horizontal or vertical) delivered through the Cloud service model (Multi-tenant, self-service provisioning, elastic scaling and usage metering or pricing) via the Internet with access via Web-centric interfaces and exploiting Web-oriented cloud architecture. The BPaaS provider is responsible for the related business function(s)." [Source: IBM MI and IPR definition bridge between Gartner and IDC, Aug 19, 2010]

Examples are processes for employee benefit management, business travel, procurement or also IT-centric processes such as software testing (where the entire testing process including testing staff is provided as an externally hosted cloud service).

2.3.2.1.2. Cloud service creation & ecosystem aspects
It is very important to understand the relationship between a cloud service and the artifacts which can be developed based on and within the boundaries of an ecosystem-focused IaaS or PaaS cloud service. Bringing any cloud service to market requires corresponding pre-investment, along with respective
metering & charging models in support of the corresponding business model. For example, Amazon EC2 is an eco-system focused IaaS cloud service. The eco-system artifacts in EC2 are VM images. EC2 allows uploading newly created VM images and charging for these VM images. However, each VM instance inherits almost all technical and business decisions already made by Amazon when they decided to take EC2 to market for a specific price-point: Each VM on EC2 is running based on the availability & performance SLAs as defined by EC2, the metrics which can be used to charge for VMs are the ones defined by EC2, the management actions (start, stop, reboot, etc.) consumers can perform on the image are the ones pre-defined by EC2, etc. The fact that Amazon nails down all these characteristics has a very good reason as each characteristic has implications on the costs for offering EC2 – therefore making the characteristics flexible to artifact developers is not possible as it would be very hard to make the corresponding costs flexible and by that very hard to predict.

This illustrates that defining and delivering a cloud service requires nailing down all corresponding functional and non-functional requirements. The artifacts developed on top of an ecosystem-focused cloud service have then only very minimal room to change how these functional and non-functional requirements are addressed. Note that this is not to be viewed as something negative, but rather as something very positive from an eco-system perspective – it is a core value proposition of eco-system focused cloud services to provide pretty strict guidelines with respect to how they can be exploited as this is the main factor driving a reduction in cost of artifact development. The easier it is to develop artifacts for such a cloud service, the more likely the cloud service is successful.

As a summary, it is important to note that there is a difference between developing cloud services as a very conscious technical and business decision vs. developing artifacts on top of eco-system focused cloud services prescribing the boundaries for how these artifacts can run.

Note that sometimes the concept of a “Cloud Service” is also referred to as a “Cloud Service Product”.


2.3.2.2. Infrastructure

“Infrastructure” represents all infrastructure elements needed on the cloud service provider side, which are needed to provide cloud services. This includes facilities, server, storage, and network resources, how these resources are wired up, placed within a data center, etc. The infrastructure element is purely scoped to the hardware infrastructure, therefore it does not include software running on top of it such as hypervisors (hypervisors are generally specific to an IaaS offering and therefore belong to that particular cloud service). Consequently it also does not include any virtualization management software.

The decision whether the infrastructure is virtualized or not depends on the actual workload characteristics to be run on the respective infrastructures: For many workloads (e.g. compute & storage as-a-Service) it is very convenient to virtualize the underlying infrastructure, especially since virtualization enables some use cases which can basically not be realized with a physical infrastructure (e.g. all use cases related to image management or dynamic scaling of CPU capacity as needed). For other workloads (e.g. analytics/search) it is required to have maximum compute capacity and use 100’s or 1000’s of nodes to run a single specialized workload. In such cases a non-virtualized infrastructure is more appropriate. As mentioned in 2.3.2.1, this is not a violation of the architectural principles postulating as much as possible commonality across cloud services: While maximum commonality is a core architectural principle, it is allowed to have different infrastructure architectures per workload category. For example, a collaboration, web and infrastructure workload requires a different underlying infrastructure than an HPC or highly transactional workload. However, a requirement in any case is that all of these infrastructure components get managed from a single, central CCMP and CCMP has the ability to place instances of each cloud service on the corresponding infrastructure (or IaaS service...
instance, in case a SaaS instance is not directly running on an infrastructure but leverages a IaaS cloud service as an alternative sourcing model).

The less variance the infrastructure has, the more it caters to the standardization needs of a cloud environment. Minimal variance on the infrastructure side is critical for enabling the high degrees of automation and economies of scale which are base characteristics of any cloud environment. However, it has to be acknowledged that in many installed cloud computing environments (specifically private clouds) there are different workloads to be provided as a cloud service and each of these workloads might have special infrastructure needs and might need to support different SLAs. So although the ideal case is total homogeneity on the infrastructure side, it is important to note that there will cloud installations with a few variants in the infrastructure elements (e.g. different HW platforms).

The infrastructure is managed by the OSS as part of the CCMP, whereas the CCMP by itself is also running on the infrastructure.

Note: The physical existence of a virtualized infrastructure on the cloud service provider side is not mandatory, since it is also possible for a cloud service provider to consume infrastructure as a service (and the required CCMP) from a different cloud service provider and put higher value cloud services on top. Clearly, the consuming service provider inherits all SLA constraints defined for the consumed cloud service. So depending on the capability to implement SLAs in software or by using other means, improving SLAs beyond what is provided by the underlying cloud service might be hard (admittedly, there are exceptions to this statement, specifically for cloud workloads which have QoS totally realized in software).
2.3.2.3. Common Cloud Management Platform (CCMP)

The Common Cloud Management Platform exposes a set of business and operational management focused services (BSS & OSS). These BSS & OSS functions must be exploited by Cloud Services to run within the context of the respective cloud service provider (and the corresponding CCMP installation). Besides OSS and BSS, the CCMP also includes User Interfaces serving the three main roles defined in the CC RA – a Service Consumer Portal to be used by Cloud Service Consumers for self-service delivery & management (the actual cloud service instances are used via a cloud service specific UI, a Service Provider Portal serving Cloud Service Provider internal users & administrators for daily operations and a Service Development Portal used by Cloud Service Creators. CCMP functionality is accessible via APIs exposed by the CCMP-internal components. Note that the architecture described in this work product is agnostic to the actual software products used to implement this architecture.

As the name already implies, the CCMP is structured as a platform. Based on the platform nature, the CCMP exposes a set of services which can (and sometimes must) be used within the context of a specific cloud service. The management services exposed by the CCMP to cloud service creators are not to be confused with the cloud services developed by cloud service creators. Cloud Service Creators are strongly encouraged to use the management services provided by the CCMP in order to enable the economies-of-scale required for achieving the extremely high degrees of efficiency associated with any cloud computing environment.
As an example, it is required to apply a special audit for any software component having financial impact, which means the component doing billing for the consumption of a cloud service must be audited to be compliant. Since every cloud service requires integration with such financial systems, the effort, cost and complexity associated with such an auditing processes can be very cumbersome. By establishing a single deployment of a billing component – shared amongst multiple cloud services – the complex and time-consuming audit process has to be executed only once and can then be used for any number of cloud services vs. executing a separate audit each time a new cloud service is deployed in an environment without a CCMP. Clearly, this concept of sharing enables economies-of-scale and does not only apply for the billing service of BSS, but for any other management service being part of a CCMP deployment.

The CCMP is defined as a general purpose cloud management platform to support the management of any category of cloud service across I/P/S/BPaaS.

The CCMP is split into two main elements – the Operational Support Services (OSS) and Business Support Services (BSS). Historically, the OSS and BSS acronyms have been used in the Telco industry in the context of providing Telco-centric services. Since cloud computing targets a similar scenario – providing services over a network – it feels appropriate to use the same conceptualizations as the Telco industry. However, the cloud services provided in the cloud computing context have a more IT centric nature (e.g. virtual machines, web conferences, etc.) than classical communication services (e.g. dialtones on a mobile phone).

In the classical telco/communication service provider world OSS and BSS are treated as operational / business support system, which implies a kind of monolithic notion of the respective component. In contrast, OSS and BSS are more viewed as a set of operational / business support (self-contained) services in the cloud computing context, which expresses a higher degree of modularity of OSS and BSS. This architectural modularization also makes it possible for cloud service creators to flexibly choose implementations of the various architectural elements being part of CCMP.

The ideal case from a cost optimization and economies-of-scale perspective is to use as much as possible shared CCMP OSS/BSS functionality across multiple cloud services, but depending on the isolation or time-to-market requirements for certain cloud services this may not be possible. In general, OSS + BSS should be viewed as the (integrated) set of management platform functionality underpinning the operation of any cloud service (similar to middleware being the functional / runtime platform).

All CCMP functions can be used for the consistent management of cloud services virtualized on any level – be it hypervisor-level, Operating-system level, platform-level or application-level virtualization (see figure below).
A portal is the interface to the external world, whether it is a client, provider or partner. In this RA, there are three different portals which can be decomposed with the same set of components. While we refer to three different portals from an architectural perspective, all three portals could be realized by a single implementation with different access rights and presenting different capabilities depending on who is logged in:

1. **Service Consumer Portal**: This allows consumer, business managers and administrator to manage their cloud environment in a self-service fashion. The Service Consumer Portal exposes information & functionality for Service Catalog, Service Instance management, User/Entitlement management and Reporting. The service consumer portal does not expose cloud service-specific usage functionality – this is covered by cloud service specific UI implementations. For example, the UI for running a web conference via LotusLive, displaying all participants, showing the screen of other participants is not something that would be implemented in the context of the service consumer portal.

2. **Service Provider Portal**: This allows service providers to manage the infrastructure and use the management components included in CCMP.

3. **Service Development Portal**: This supports cloud service creators in creating new cloud services.

Similar to OSS & BSS, the Service Consumer UI/Portal components are split into a platform-/cloud service independent part and cloud service-specific UI definitions, which can be plugged into that base framework.
2.3.2.3.1. BSS – Business Support Services

Business Support Services represents the set of business-related services exposed by the CCMP, which are needed by Cloud Service Creators to implement a cloud service.

The motivations for using these commonly defined BSS functions are the same as the ones for reusing all other services defined in the context of CCMP:

- Using standard component services will reduce the unique development required, reducing development and support costs both
- Using standard components will result in a cloud service more likely to be compatible with other cloud services, providing the client with even more economies of scale and efficiencies
- Standard components and User Interfaces will reduce the complexity, simplify operations and make the cloud solutions easier to use (more consumable)

Like any other component of the CCMP, the BSS is generic across all cloud service types and can be configured to behave appropriately in the context of the managed cloud services. As an example, the billing service of the CCMP BSS must be usable to do billing for the consumption of virtual machines...
(IaaS), a multi-tenancy capable middleware platform and for collaboration services like LotusLive (SaaS). This drives the need for a proper platform-level definition of all BSS components and exploitation artifacts enabling cloud service creators to prime the behavior of each BSS component in a cloud service specific fashion.
2.3.2.3.2. OSS – Operational Support Services

Operational Support Services represents the set of operational management / technical-related services exposed by the CCMP, which are needed by Cloud Service Creators to implement a cloud service.

Many management domains shown in the OSS can also be encountered in traditionally managed data centers (e.g. monitoring & event management, provisioning, incident & problem management, etc.) while other components are new and pretty specific to the degrees of automation and efficiency associated with clouds (e.g. service automation, image lifecycle management). Particularly for the ‘traditional’ management domains it is important to note that conceptually they are the same in the cloud world and in the traditional world, whereas in a cloud world these domains are generally implemented in radically different ways taking advantage of the high degrees of homogeneity in a cloud. For example, a traditionally managed data center is implemented in a way that an incident gets raised if a physical server fails, a ticket gets opened, assigned to an admin (maybe 2 AM in the morning), after some time an escalation takes place if the admin hasn’t resolved the ticket until then, etc. In contrast, there is also incident & problem mgmt for typical VMaaS cloud services, whereas here a broken physical machine is just left broken on the floor until some later point of time and the virtual machines which have been running on that physical machine are brought up on another one. Of course, this is only possible if the availability SLA associated with the VMaaS cloud service permits such approaches. Both scenarios described above address incident & problem management, but in radically different ways and for radically
different labor costs and SLAs. A similar “cloudified” perspective exists also for most other OSS components.

The platform notion of CCMP obviously also applies to all components defined as part of the OSS: A proper platform-level definition of all OSS components and exploitation artifacts enabling cloud service creators is needed to prime the behavior of each BSS component in a cloud service specific fashion.

### 2.3.2.4. Security, Resiliency, Performance & Consumability

![Diagram](image)

**Figure 9: CC RA – Security, Resiliency, Performance & Consumability Details**

Security, Resiliency, Performance & Consumability are cross-cutting aspects spanning the CCMP, the hardware infrastructure and Cloud Services. These non-functional aspects must be viewed from an end-to-end perspective including the structure of CCMP by itself, the way the hardware infrastructure is set up (e.g. in terms of isolation, network zoning setup, data center setup for disaster recovery, etc.) and how the cloud services are implemented.
2.3.3. Cloud Service Creator

2.3.3.1. Service Development Tools

Service Development Tools are used by the Cloud Service Creator to develop new cloud services. This includes both the development of runtime artifacts and management-related aspects (e.g. monitoring, metering, provisioning, etc.). "Runtime artifacts" refers to any capability needed for running what is delivered as-a-service by a cloud deployment. Examples are JEE enterprise applications, database schemas, analytics, golden master virtual machine images, etc.

In the context of a particular infrastructure- or platform-as-a-service offering, there may also be tooling to develop artifacts which are specific to the particular cloud service. For example, in the context of a VM-as-a-service offering, it is possible to use image creation tools for developing images that can be deployed in the context of the VM-aaS cloud service. As another example, in the context of a platform-aaS cloud service there may be application development tooling to develop an application which can be deployed on the respective platform.
3. CC Reference Architecture: Architectural Principles and Related Guidance

The following top-level architectural principles guide the definition of any cloud implementation, with a focus on delivery & management of cloud services.

The architectural principles in this chapter are focused on the CCMP element of the overall architecture as this element is required consistently, independent of which cloud service is implemented, delivered & managed:

1. **Design for Cloud-scale Efficiencies**: When realizing cloud characteristics such as elasticity, self-service access, and flexible sourcing, the cloud design is strictly oriented to high cloud scale efficiencies and short time-to-delivery/time-to-change. (“Efficiency Principle”)


3. **Identify and Leverage Commonalities**: All commonalities are identified and leveraged in cloud service design. (“Economies-of-scale principle”)

4. **Define and Manage generically along the Lifecycle of Cloud Services**: Be generic across I/P/S/BPaaS & provide ‘exploitation’ mechanism to support various cloud services using a shared, common management platform (“Genericity”).

These principles can be summaries as ELEG (= Efficiency, Lightweightness, Economies-of-scale, Genericity).

It is the objective of this chapter to refine these top-level principles into memorable and actionable guidelines for cloud architects and developers.

### 3.1. CC RA-Specific Architectural Principles

#### 3.1.1. Origins of the Term Architectural Principle

The notion of architectural principles is commonly used in the enterprise architecture community. The Open Group Architecture Framework (TOGAF), for instance, provides a definition [1]. There is defined: Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission. In their turn, principles may be just one element in a structured set of ideas that collectively define and guide the organization, from values through to actions and results.

Depending on the organization, principles can be defined and established on different levels. Architecture principles are a subset of IT principles that relate to architecture work. They reflect a level of consensus across the enterprise, and embody the spirit and thinking of the enterprise architecture.

In contrast to an architectural decision, an architectural principle is an overarching guideline or paradigm driving architectural decisions on a more granular level (which reflects the enforcement of an architectural principle). Architecture standards as well are more granular than architecture principles and describe technical specifications.
3.1.2. Principle 1: Design for Cloud-Scale Efficiencies

The first CC RA principle is the overarching policy governing the architecture, design, and implementation efforts:

**Architectural Principle 1 (“Efficiency Principle”):**

> When realizing cloud characteristics such as elasticity, self-service access, and flexible sourcing, the cloud design is strictly oriented to high cloud scale efficiencies and short time-to-delivery/time-to-change.

**Motivation and Benefits**

The objective of cloud computing in general and CC RA in particular is to enable changes in orders of magnitude with respect to operational efficiencies (compared to traditional IT environments); hence, any design activity must be directed with this top-level principle and policy in mind.

Operational and maintenance efforts and costs with today’s IT landscape are too high. The complexity and interdependencies within the IT landscape is too high. IT operations are by far too much based on manual activities and procedures. The time needed to change an existing IT system or to provide a new IT system is longer than the business now can accept.

**Consequences and Implications**

To really implement a cloud following this principle with that high level of efficiency and flexibility, a very high degree of standardization (i.e. minimal variety in the data center with respect to numbers of server, storage, network technologies, operating systems & versions, middleware products, applications, etc.) is required to enable high degrees of automation. The higher the degree of standardization / minimization of variety is, the better automation can be realized (assuming a well-integrated and interoperable set of management components). Obviously, in a highly homogeneous public cloud data center this can be achieved in a better way compared to private cloud enterprise data centers running a variety of workloads each of them having different requirements, so there is typically a trade-off between degree of standardization and level of efficiency.

All IT services have to be defined with the highest degree of standardization and documented in a comprehensive IT service catalogue.

For each cloud service every service request has to be described and implemented to be run on an automated basis.

A cloud management platform has to be designed and implemented that is able to manage the status of every operational task and all (typical known) operational deviations (aka incidents).

**Examples**

Infrastructure costs can be significantly reduced if their utilization is increased. That is quite easy to be done by leveraging the degree of standardization of the infrastructure components as well as the whole IT services. The IT services are run on a standardized and shared IT environment that keeps IT costs low.

Deliveries as well as change tasks for example today are too often based on manual work and coordination. In a cloud environment most of these typical tasks can be automated; manual work is only necessary in case of exceptional situations. Keeping manual effort and therefore labor costs low is achieved by a high level of automation.
3.1.3. Principle 2: Support Lean Service Management

Cloud computing will only be able to deliver on its promises if a paradigm shift for data center and service management design takes place, including a redesign of management processes and related tools. This principle is an implication of principle 1:

**Architectural Principle 2 ("Lightweightness Principle"):**

*The Common Cloud Management Platform fosters lean and lightweight service management policies, processes, and technologies.*

Motivation and Benefits

Today’s service management processes are by far too much based on manual and individual work and far away from being lean and automated.

Current service management technologies are not integrated and comprehensive enough to address all global service management functions.

Consequences and Implications

All (automated) service management functions need to communicate and deal with status information.

An overall status and steering function is necessary.

At least a monitoring function on status information is necessary.

The starting point as well as finishing the service delivery should be related to self-service functions.

The operational teams need other skills than today esp. a higher skill level on cloud service delivery.

Existing implementations of IT Service Design (as of ITIL v3) might have to be re-designed to deliver IT services conform to the CCMP.

New technical service management components (i.e. beyond traditional enterprise data center management) are required in CCMP implementations, e.g., service automation management or image lifecycle management. Not all Commercially-Of-The-Shelf (COTS) products e.g. might be able to support CCMP as defined in the CC RA without customization or extensions. In those cases it is critically important to understand how the existing management tools have to be used in novel/different ways to achieve cloud-scale efficiencies. Technical variability should be limited so that management can be streamlined and automated as much as possible. The degree of achievable efficiency also depends on interoperability & integration amongst the set of management tools being used.

Examples

One frequently cited example is the incident, problem, and change management approach taken by several public cloud implementations.

Let us take as an example that a cloud service consumer reports that a virtual machine has gone down and that the root cause of the problem is a hardware failure. In the traditional IT Infrastructure Library (ITIL) approach to change management, the help desk would alert a technician who would repair the faulty hardware as soon as possible. This guarantees a high quality of service (here: time-to-repair), but is rather costly. In a cloud scenario, the broken hardware/hypervisor would just be disabled and when users...
start up their virtual machine again (because it failed due to the broken hardware/hypervisor) they are
brought up on one of the remaining healthy hypervisors. All faulty hardware is replaced or repaired on a
regular base (e.g., monthly) vs. repairing it immediately whenever it fails. Due to the higher amount of
automation, less admin staff is required to provide the service. This is an example for dramatically
increasing the efficiency in operating the data center, which should serve as an inspiration for novel ways
of dealing with old problems, leveraging new technology approaches and building on high degrees of
standardization and automation.

Note that this example shouldn’t be viewed as the ‘authoritative’ way of dealing with hardware failures in
all cloud implementations: Of course, depending on the SLAs of the actual workload running on the cloud
and the underlying hardware it might well be that a different approach for dealing with such problems is
more appropriate.

Fundamental restructuring and streamlining of IT management processes is required to maximize the
elimination of manual data center management tasks. Depending on the degree of optimization, this
might also require deviating from well-established standards for traditional data center management (e.g.
ITIL).

This principle is not about gradual improvements of data center management processes as it is driven by
many data center optimization initiatives. The reason for this is that with such an approach of gradual
improvement it would only be possible to achieve gradual improvements of 10-15%, which does not fit to
the order-of-magnitude improvements as they are targeted from the perspective of cloud-like data center
management. Along these lines, this principle is also not about the 1:1 automation of existing data center
management processes, as this would mostly mean transforming existing heavyweight processes into
automated heavyweight processes, whereas actually a fundamental shift in data center management
processes is required. Such a paradigm shift can be achieved by using most of existing service
management tools (enriched by a few, new & distinguishing tools), but they must be used in very different
ways.

The main levers for massively driving down operational costs are elimination of tasks which are not
needed any more due to limited scope of managed (e.g. in compute cloud only managing up to and
including the hypervisor) and optimization (e.g. not immediate repair of a failed physical machine enabled
by automated restart of crashed VMs or nodes or automating the service activation process, which is
typically very time- and cost-intensive in traditional outsourcing environments). The basic thought is the
more tasks get eliminated, and the more homogenous a data center is, the easier it becomes to highly
automate any type of manual management task and drive down mgmt costs & delivery/change times
significantly as a result of that.

Note that even for scenarios where management scope is not reduced (i.e. mgmt is still done up to the
application layer), it is possible to eliminate management tasks by leveraging the sourcing options offered
via cloud computing.

For example, a cloud service provider offering SaaS can eliminate his management tasks for the
underlying hardware infrastructure or for maintaining the required management software by retrieving the
compute resources or the management functionality as-a-Service. Consequently the corresponding labor
costs get eliminated on the SaaS-provider side while he has to pay the compute or mgmt-aas provider for
the respective service. However, since the underlying service provider is able to specialize in providing a
very specific service and can distribute costs for providing that across multiple tenants (economies of
scale), the overall costs for the SaaS-provider might well come down. Of course, this also works within a
cloud service provider environment where a higher-level service can build on a lower-level service,
management being highly optimized for each of them.
3.1.4. Principle 3: Identify and Leverage Commonalities

The third principle reminds us that the first ‘C’ in CCMP does not stand for ‘Cloud’, but for ‘Common’:


All commonality are identified and leveraged in cloud service design.

Motivation and Benefits

Although various and a lot of business as well as technical functions occur in the IT landscape, too many parts are implemented and operated on an individual, separated way. So redundancies that drive complications (as well as complexity) are daily business in the IT. Even some activities to identify commonalities are not established and use within the whole IT organization.

Hence, the motivation for this principle is to reuse management/CCMP components and enable economies of scale (with respect to initial standup & operational costs and reduced time-to-market) by sharing a single/common management platform to deliver and manage many cloud services. An exploitation mechanism is required so that various cloud service types – across I/P/S/BPaaS (see also principle 4, section 3.1.5) – can share the same common management platform while having cloud service specific management requirements and scope.

Consequences and Implications

A more generic design requires more analysis, design, and development effort; it runs the risk of over-engineering (if the expected future requirements and usage scenarios never materialize) and compromising usability (if the responsibility to configure the feature for a specific usage scenario is shifted to the service consumer or the system administrator and the configuration procedure is time consuming and/or difficult to follow). This can be counter productive, particularly in the light of the overarching CC RA principle 1, design for cloud-scale efficiencies.

Another consequence is, that a uniform management API becomes enabled for interactions with different service types and instances.

Examples

Metering, monitoring, service automation, management User Interface (UI) components are examples of components with reuse potential – just like any other component which is defined as part of CCMP (if a component cannot be split into a platform aspect and an exploitation aspect, it probably doesn’t make sense to make it part of CCMP, but treat it as something cloud service specific).

A counter example is the connection broker in the Desktop Cloud offering; Desktop connections do not exist in any other cloud offering type, so there is not much commonality potential in the design of this particular connection broker (as this component will hardly be reused for other cloud services). Therefore the connection broker would be handled as a management component residing outside the Common Cloud Management Platform, but within the specific context of a Desktop Cloud Service.
3.1.5. Principle 4: Define and Manage Cloud Services generically along their Lifecycle

A top-level architectural principle for the CC RA requires us to build on the notion of a service lifecycle as a central thought:

Architectural Principle 4 (“Genericity Principle”):
Cloud services are managed generically along their lifecycle, across I/P/S/BPaaS.

Motivation and Benefits

IT services today are usually not designed generic but specific. This is a large hurdle for reusing any service element in another IT service.

A CCMP as defined in this RA is supposed to scale and to host multiple types of cloud services (across I/P/S/BPaaS), so it is essential to introduce a model which allows cloud service developers to specify how the CCMP functionality gets used in the context of their specific cloud service and how – based on that definition or template – instances of that cloud service get delivered to cloud service consumers (in line with the class-instance thinking from OO). This has been taken into account in the CCMP RA design. The remaining todo for the cloud management platform architect is to design how the CCMP management functionality gets exploited in the context of the respective cloud service. This is achieved by creating a set of service type-specific artifacts as required by the respective management platform components (e.g. cloud service-specific scripts, monitoring agents, etc.).

It is important to be specific about the semantics of the term ‘service’ when using it in the cloud context as just talking about the term ‘service’ is too imprecise. In the context of the CC RA, the notion of a service and of a service instance exists, whereas a service is responsible for creating service instances and represents their runtime context. Services instances service as defined in the CCMP RA are typically not “classical” Web services; neither one of these concepts corresponds to an ITIL service either. In the context of the CC RA, service instances represent tangible units which are referred to in noun form (they are the units of delivery and management). Such cloud service instances represent something that is provided “as-a-service” (going across all aaS categories), such as an infrastructure element like a virtual machine, a middleware platform or an end user application. All these entity-like service instances have a lifetime that begins with deploying and delivering them; it ends with deprovisioning them. As a rule of thumb, everything that may appear in the XaaS stack can be a service instance.

Consequences and Implications

The cloud service creator needs to have a thorough understanding of the functionality provided by the CCMP and how this functionality can be leveraged within the context of the respective cloud service.

All IT services have to be clearly described due to their lifecycle. For every IT service a set of artifacts has to be created in a consistent fashion, according to the guideline defined in the CC RA “cloud service creation” work product.

Examples

One example is capturing the specifics of providing and managing storage-as-a-service. As part of creating a storage cloud service all artifacts required for delivering & managing a new storage service instance are created. That means, in the context of a storage-aaS cloud service the cloud service consumer gets a file system or a storage volume to put data on.
4. References

[3] Amazon, AWS and EC2 resources
[6] IBM GTW world-wide CoP lecture series (contact: Teisha Harry)