

Fig. 2.1 Pattern map of cloud computing fundamentals

In this chapter, we introduce the fundamentals required for the understanding of the following chapters. As stated in the introduction, the cloud computing properties – access via network, on-demand self-service, measured service (pay-per-use), resource pooling and rapid elasticity – fundamentally change how IT resources are provided and used. It is important to understand why cloud offerings have these properties, how these properties are delivered on different levels of a typical application stack and under which conditions an application benefits from them. We begin by examining application workloads (Sect. 2.2) and show how they influence the decision for the adoption of cloud offerings. Especially, we discuss how applications experiencing different types of workloads can benefit from the cloud computing properties covered in Chap. 1. As in that previous chapter, we use

All figures published with kind permission of © The Authors 2014. See list of figures.

the NIST cloud definition [3] and emphasis on those aspects that are important to understand the following chapters.

Having motivated the need for cloud offerings to handle different workloads we introduce common cloud service models (Sect. 2.3) that describe different styles to offer IT resources on different levels of an application stack. We cover the layers of this application stack and their function in an application. Furthermore, we discuss how the corresponding service models *Infrastructure as a Service*, *Platform as a Service* and *Software as a Service* enable the cloud computing properties. In the last section of this chapter, we introduce the cloud deployment models (Sect. 2.4) and describe how they differ regarding the sharing of IT resources, reaction to varying application workloads, economies of scale, and costs. These properties significantly affect cloud adoption in companies influenced by the concern that outsourcing parts of an application stack and sharing IT resources with other companies can negatively impact privacy and security of data and processes.



Side Note: the patterns in this chapter differ from other patterns in this book. They are not implemented by developers, but characterize the context in which other patterns are applicable. We used the pattern format to correlate them with other patterns in a uniform way and use their icons to characterize cloud environments and application requirements in Chap. 7.

2.1 Overview of Fundamental Cloud Computing Patterns

As seen in Fig. 2.1, the map of patterns covered in this chapter for application workloads, cloud deployment types, and cloud service models are strong interconnected. While this is not the case for most of the other patterns in this book, these fundamental patterns form the basis for the understanding of the remaining patterns and should always be considered completely when designing cloud applications.

Patterns for application workloads (Sect. 2.2) describe different user behavior resulting in changing utilization of IT resources hosting an application. This workload can be measured in the form of user requests, processing load on servers, network traffic, amount of data stored etc. In detail, we cover *static workload* (26) that only changes minimally over time, *periodic workload* (29) that has recurring peaks, *once-in-a-lifetime workload* (33) that has a peak once, *unpredictable workload* (36) that changes frequently and randomly, and *continuously changing workload* (40) that grows or shrinks over time.

Once the workload experienced by an application can be described and categorized, it is important to understand the cloud service model (Sect. 2.3) used by a cloud provider. It affects the pricing model of providers and, therefore, how workload should be measured and evaluated in applications. We cover the three

NIST cloud service models [3] in a pattern format. *Infrastructure as a Service (IaaS)* (45) describes how servers are offered by cloud providers. *Platform as a Service (PaaS)* (49) covers cloud offerings providing complete execution environment for a specific type of applications, i.e., those developed in a certain programming language. *Software as a Service (SaaS)* (55) describes how complete applications can be offered to customers.

Cloud service models and cloud deployment models are two viewpoints on the cloud provider: cloud service models describe the style how IT resources are offered. Cloud deployment models describe the cloud environments hosting these IT resources, especially, regarding the group of customers they are made available to. Therefore, a combination of cloud service model and cloud deployment type characterizes the environment of a cloud provider. For each of the cloud deployment model patterns we describe the combinations with cloud service model patterns in the related patterns section and discuss the usage scenarios for each combination. The covered cloud deployment models are as follows: A *public cloud* (62) is generally available to everyone. A *private cloud* (66) is hosted exclusively for one company. A *community cloud* (71) is a cloud environment between these two extremes and is made accessible only to a certain group of companies or individuals that trust each other and often wish to collaborate. Finally, *hybrid clouds* (75) provide means to interconnect clouds of the other deployment models to distribute applications among various hosting environments.

2.2 Application Workloads

We use the term workload to refer to the utilization of IT resources on which an application is hosted. Workload is the consequence of users accessing the application or jobs that need to be handled automatically. Workload becomes imminent in different forms, depending on the type of IT resource for which it is measured: servers may experience processing load, storage offerings may be assigned larger or smaller amounts of data to store or may have to handle queries on that data. Communication IT resources, such as networking hardware or messaging systems may experience different data or message traffic. In scope of the abstract workload patterns, we merely assume this utilization to be measurable in some form. These measurements form the basis to increase or decrease the number of IT resources assigned to an application during elastic scaling, one of the cloud application properties introduced in Sect. 1.2 on Page 5.

From a customer perspective the desire to only pay for the IT resources that are actually used is common across many outsourcing domains – whether they are IT or not. A non-IT example of this desire is an airplane ticket – the customer of the transport capability only pays for the exact flights he or she takes and does not need to buy a plane upfront, train the pilots and deal with the maintenance of the plane afterwards just to take a flight. The requirement to reduce up-front capital expenditures (CAPEX) and move costs to operational expenditures (OPEX) that grow and shrink with the actual consumption of a service has led to the adoption of

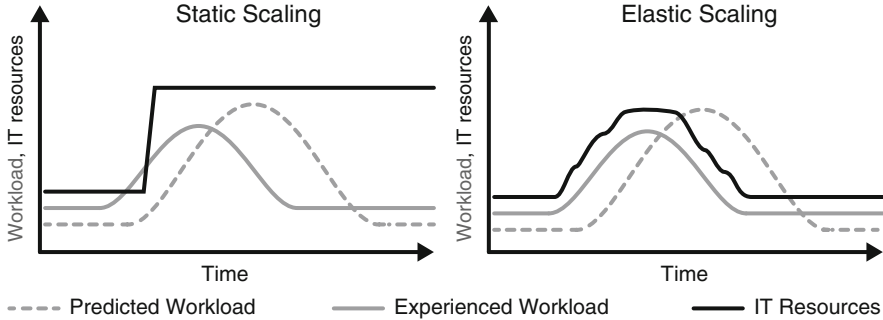


Fig. 2.2 Exemplary resource provisioning

pay-per-use as one of the fundamental properties of cloud offerings. As customers desire to pay for used resources only, providers must employ the principle of rapid elasticity to elastically grow or shrink the resources assigned to a customer based on that customer's demand. Therefore, at least two of the essential cloud properties – pay-per-use and rapid elasticity – result from the demand to cope with non-static application workloads. Varia [28] discusses the total cost of ownership in detail for handling different workloads.

In the following we examine some common utilizations of IT resources over time. This *workload* is the result of user requests to an application or cloud offering resulting in processing load, communication traffic, or data to be stored. The workload patterns discussed in this section cover different types of workloads we found in literature and observed in applications. For each of these workload patterns, we discuss how they can be handled efficiently in a cloud environment.

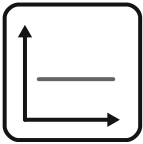
Figure 2.2 shows a general problem that arises in scope of workload changes to which a scaled-out application has to react by changing resources numbers. Whenever workload is predicted as shown by the *predicted workload* curve, it may be experienced slightly different, as shown in the *experienced workload* curve. In this example, a predicted peak in workload started a little earlier than expected and resource numbers have to be adjusted accordingly.

In case of static scaling, where physical servers are provisioned, the time it takes to order, setup and start them may not be reactive enough to handle the faulty prediction. Therefore, the necessary resources become available too late resulting in an “*underprovisioned*” application. To cope with the inflexibility of such resources, they have to be provisioned for the predicted peak-load right from the beginning and are hard to decommission once the workload decreases. This results in an “*overprovisioned*” application after the peak. This over- and underprovisioning has a direct impact on the properties of the hosted applications. An underprovisioned application usually cannot provide the desired user experience, as performance is reduced and, for example, reactivity of the application decreases. Overprovisioning has a lesser impact on the user of the application, but leads to higher costs as resources are provisioned but remain unused.

Elastic scaling depicted on the right of Fig. 2.2 can provision and decommission resources much more flexibly and, thus, is not as dependent on workload predictions. Once the increase is detected, new resources are provisioned in small intervals and this provisioning is stopped even though the predicted workload peak has not been reached. Therefore, elastic scaling allows a much tighter alignment of IT resource numbers to experienced workloads, but has to be respected by the application architecture. In consequence, when an application is deployed in the cloud and experiences some of the workload patterns covered in this section – it has to be built in a way underlying IT resources can be added and removed. Now, we discuss the workload patterns, how they benefit from cloud properties, and point to entry points in the following chapters where the necessary application architectures enabling these benefits are covered.

2.2.1 Static Workload

IT resources with an equal utilization over time experience static workload.



How can an equal utilization be characterized and how can applications experiencing this workload benefit from cloud computing?

Context

Static workloads are characterized by a more-or-less flat utilization profile over time within certain boundaries. This means that there is normally no explicit necessity to add or remove processing power, memory or bandwidth for change in workload reasons. When provisioning for such a workflow the necessary IT resources can be provisioned for this static load plus a certain overprovisioning rate to deal with the minimal variances in the workload. There is a relatively low cost overhead for this minimal overprovisioning.

Solution

An application experiencing *static workload* is less likely to benefit from an elastic cloud that offers a pay-per-use billing, because the number of required resources is constant. The elasticity of a cloud environment is not required to handle *static workload*, but applications may still benefit from shorter IT resource provisioning times in case of resource failures or during maintenance. Homogenization of these resources, an effect of the cloud computing property *resource pooling* introduced on Page 4 in Sect. 1.1 also reduces the complexity of the runtime environment and may enable applications to be developed and deployed more quickly.

Figure 2.3 depicts an exemplary *static workload* handled using static scaling (left) and elastic scaling (right). Both provisioning approaches have the same amount of overprovisioned resources in case the experienced workload is lower than the predicted workload. In scope of elastic scaling, small adjustments may be made if the experienced workload comes close to utilizing the IT resources completely. However, the benefits of such dynamic adjustments are very limited as *static workload* does not change at all or only very little over time.

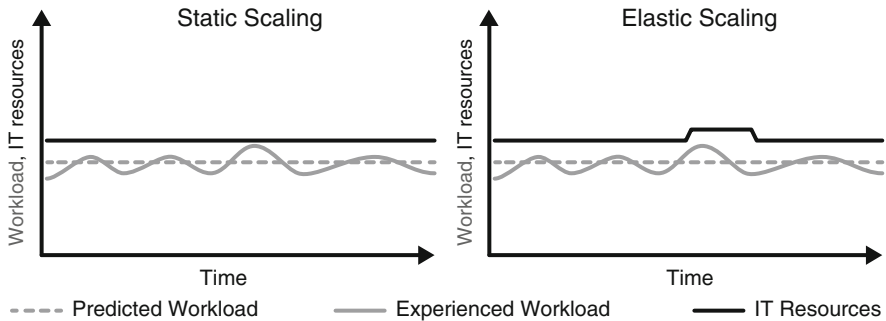


Fig. 2.3 Exemplary static workload

Result

An elastic cloud may be beneficial even for *static workloads*, because elasticity does not only provide costs savings. As clouds can provision new resources very quickly, often, within minutes, elasticity also simplifies provisioning and decommissioning tasks that are necessary for other reasons, for example, to address resource failures or for maintenance purposes. In case resources fail or need to be taken down for maintenance, an elastic cloud would, therefore, enable the rapid provisioning of new resources to continue normal operation. Moving *static workloads* to a cloud offering can also be beneficial because of the IT resource homogenization. As the provider of the cloud offering can exploit economies of scale, the use of a standardized cloud offering might be cheaper than a build-your-own solution even when the workload transferred to the cloud offering is static.

In conclusion, the cost benefits of a cloud offering might be limited or non-existent in case of *static workload*. Costs may even increase. However, in certain cases the effects of homogenization of IT resources and elasticity in failure cases may still provide the necessary benefits to use a cloud.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): this cloud service model describes how (virtual) resources, such as servers, may be provided by a cloud. An application experiencing *static workload*, requires a fixed number of such servers, but may still benefit from such an environment if the provider supplies ready-to-use server configurations. Also, a server failure may be easier to cope with using provider-supplied management functionality.
- *Platform as a Service (PaaS)* (49): according to this cloud service model, the provider offers an environment to which custom developed applications are deployed directly. This alleviates the application developer from maintaining the hosting infrastructure required by the application. Elastic scaling of hosted application can also be handled by the provider transparently to the customer,

however, this is not the main benefit for applications experiencing static workload.

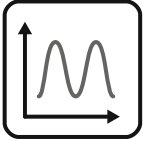
- *Software as a Service (SaaS)* (55): if complete applications experiencing static workload are obtained from a cloud environment, customers commonly pay on a per-user basis and benefit from provider-supplied infrastructure management.
- *Cloud deployment types* (Sect. 2.4): these patterns should be considered during the selection of a cloud provider to ensure that the application's requirements regarding security, privacy, and trust are met.
- *Elastic infrastructure* (87) and *elastic platform* (91): these patterns describe the how cloud offerings providing the runtime for the application behave, what interfaces they provide and which application artifacts have to be provided by developers. The interfaces are what an application experiencing static workload would interact with to provision replacements for failing resources as mentioned above.
- *Watchdog* (260): this pattern describes how applications may be monitored for failures. How to react to them automatically is further covered by the *resiliency management process* (283) that is executed by a *watchdog*.
- *Update transition process* (275): this pattern describes how the elasticity of a cloud may be used to switch between application versions. As this necessity is independent of the workload experienced by an application, is it also useful for applications experiencing *static workload*.

Known Uses

Static workload is experienced by many applications that do not fully utilize a single (virtual) server. Examples are private Websites or Websites of small and medium sized enterprises. Many small applications used internally by companies, are used continuously by a smaller user group, i.e., a documentation wiki used by one department or development group and, therefore, also experience *static workload*. Such small applications often cannot benefit from elasticity as it is impossible to shut them down during non-utilization, because their implementation was not designed for it. Also, these applications may be used periodically throughout a day hindering their shutdown as well.

2.2.2 Periodic Workload

IT resources with a peaking utilization at reoccurring time intervals experience periodic workload.



How can a periodically peaking utilization over time be characterized and how can applications experiencing this workload benefit from cloud computing?

Context

In our real-lives periodic tasks and routines are very common. For example, monthly paychecks, monthly telephone bills, yearly car checkups, weekly status-reports, or the daily use of public transport during rush-hour, all these tasks and routines occur in well-defined intervals. They are also characterized by the fact that a lot of people perform them at the same intervals. As a lot of the business processes supporting these tasks and routines are supported by IT systems today, there is a lot of periodic utilization that occurs on these supporting IT systems.

The problem with periodic tasks and static scaling of IT resources is that there must be enough IT resources to handle the utilization peaks while during the non-peak times these resources are unused. This *overprovisioning* results in a low average utilization of the allocated IT resources.

Solution

From a customer perspective the cost-saving potential in scope of *periodic workload* is to use a provider with a pay-per-use pricing model allowing the decommissioning of resources during non-peak times. This has the effect that the customer does not pay for the resources during these times. Cloud providers enable this elastic use by offering IT resources to a large group of customers that displays versatile workload behavior, a strategy that leads to the resource pooling property of clouds described on Page 4 in Sect. 1.1. Therefore, IT resources not needed by one customer are used by different customers resulting in a leveled overall utilization of the cloud offering.

Figure 2.4 shows the handling of *periodic workload* using static scaling (left) and elastic scaling (right). Static scaling always provisions IT resources for the predicted peak load regardless of the experienced workload. Elastic scaling enables the monitoring of the experienced workload. If this workload increases, new IT

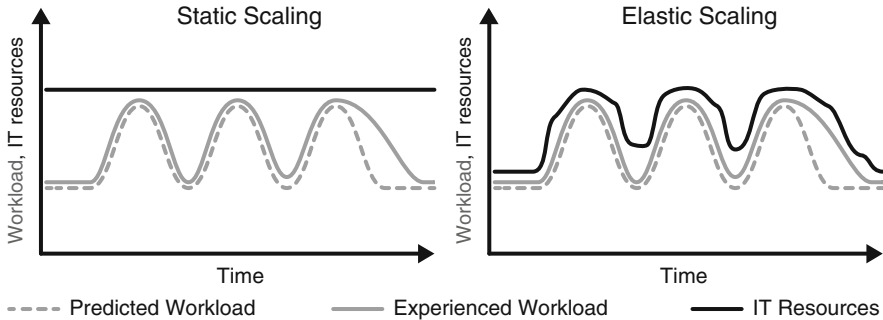


Fig. 2.4 Exemplary periodic workload

resources are provisioned dynamically. This makes the approach also less dependent on workload prediction. If the experienced workload deviates from predictions as is the case for the last peak in Fig. 2.4, monitoring detects this abnormality and resources are provisioned or, in this case, decommissioned accordingly.

Result

The benefits for customers result from unneeded resources being decommissioned during non-peak times and, thus, these resources not generating costs during these times. While the pay-per-use pricing model enables the customer to pay for used resources only, the provider still has to provision static resources to host the offering. To solve this discrepancy, the provider uses the resource-pooling cloud computing property described on Page 4 in Sect. 1.1. The resources not used by one customer can now be assigned to another customer. Thus, the business case for the provider is based on the fact that multiple customers of the cloud offering have workload patterns that complement each other in a way that the combined workload is more-or-less *static workload* (26) on the provider side. In a *public cloud* (62) offering, this static combined workload is achieved by making the cloud offering generic enough and let very diverse customers use the cloud offering. In a *private cloud* (66) the setting must be carefully evaluated so that multiple users, which can be different applications, departments or local-business units, have a combined workload that is somewhat static.

Some cloud providers, furthermore, motivate a static long term use of their offerings. For example, Amazon offers virtual servers of its Elastic Compute Cloud (EC2) [18] at lower prices if they are provisioned for longer time periods, an option called reserved instances [29]. Under such conditions, application developers should consider provisioning some IT resources required by their custom applications in a static fashion even when obtaining them from an elastic cloud. Therefore, customers may face similar challenges as the cloud provider who has to provision static resources for the peak-load of his or her offering. Due to the above

mentioned static pricing models of some *public clouds* (62), the same considerations can also lead to cost reductions when using *public clouds*.

Related Patterns

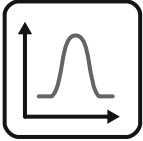
- *Infrastructure as a Service (IaaS)* (45): clouds of this service model provide (virtualized) servers on demand. To handle *periodic workload*, servers can, therefore, be provisioned and decommissioned as needed. The integration of these servers into an application has to be enabled by the application architecture as covered by the patterns discussed in Chap. 4.
- *Platform as a Service (PaaS)* (49): this cloud service model provides a hosting environment for custom applications that are deployed directly to it. Application management, i.e., to handle scaling or failure resiliency can be offered by the provider as well. Therefore, IT resource provisioning and decommissioning during the peaks of *periodic workload* may be handled transparently to the customer. However, provider-supplied management functionality often has to be configured, for example, regarding the intensity at which resources are provisioned when a workload increase is monitored. These configurations should be evaluated carefully in scope of *periodic workload* to ensure that provisioning and decommissioning offered by the provider is reactive enough for the concrete application scenario.
- *Software as a Service (SaaS)* (55): complete applications offered as a service are commonly billed per user. Pricing models can be, for example, on a per-month basis. This can hinder the effective use of such clouds in scope of *periodic workload* if the periodic peaks occur too frequently. For example, if a *SaaS* application is billed per user and month, peaks that are monthly or more frequent hinder efficient use of the offering.
- *Public cloud* (62), and *community cloud* (71): *periodic workloads* are good candidates to be outsourced to a cloud as they benefit from elasticity and pay-per-use pricing. This elasticity is most likely to be supported by *public clouds* and *community clouds*.
- *Private cloud* (66): one important aspect of *periodic workload* is that is often predictable. As the peaks are not unforeseen, the provider can plan with these peaks and try to find suitable other workloads with peaks that level-out the low-utilization times of customers. Thus, *periodic workloads* are good candidates for a resource-restricted *private cloud* where a provider can level out different users with *periodic workloads*.
- *Elasticity manager* (250), *elastic load balancer* (254), and *elastic queue* (257): to determine when to provision or decommission resources, the workload experienced by an application has to be monitored. These three patterns describe how additional management functionality can be integrated with the remainder of an application. They can be implemented best, if the application follows certain architectural styles that we cover in Chap. 4.

Known Use

One of the services provided by T-Systems to the T-City Friedrichshafen [30] is an online kindergarten signup application, where parents may sign up their children for kindergarten places available throughout the city. This sign up is only open twice per year, which is why the workload experienced by the application follows the *periodic workload* pattern.

2.2.3 Once-in-a-Lifetime Workload

IT resources with an equal utilization over time disturbed by a strong peak occurring only once experience once-in-a-lifetime workload.



How can equal utilization with a one-time peak be characterized and how can applications experiencing this workload benefit from cloud computing?

Context

As a special case of *periodic workload* (29), the peaks of periodic utilization can occur only once in a very long timeframe. Often, this peak is known in advance as it correlates to a certain event or task. Even though this means that the challenge to acquire the needed resources does not arise frequently, it can be even more severe. The discrepancy between the regularly required number of IT resources and those required during the rare peak is commonly greater than for *periodic workloads* (29). This discrepancy makes long term investments in IT resources to handle this one-time peak very inefficient. However, due to the severe difference between the regularly required IT resources and those required for the one-time peak, the demand can often not be handled at all without increasing IT resources.

Solution

The elasticity of a cloud is used to obtain IT resources necessary to handle the *once-in-a-lifetime workload* flexibly. The provisioning and decommissioning of IT resources as well as their use and integration in an existing application can often be realized as a manual task performed by humans, because it is performed very rarely at a known point in time.

In Fig. 2.5, an exemplary *once-in-a-lifetime workload* is depicted. Characteristic for this type of workload is the large difference between the IT resources required during the peak and those required otherwise. As for *periodic workload* (29), static scaling provisions IT resources for the predicted workload peak. If this prediction is wrong and the experienced workload is higher, as is the case in Fig. 2.5, additional resources often cannot be provisioned quickly enough affecting the performance of the application and possibly the user experience. On the right side of Fig. 2.5, the same workload is handled using elastic scaling. As IT resources are provisioned manually for the one-time peak the IT resource curve shows sudden increases and

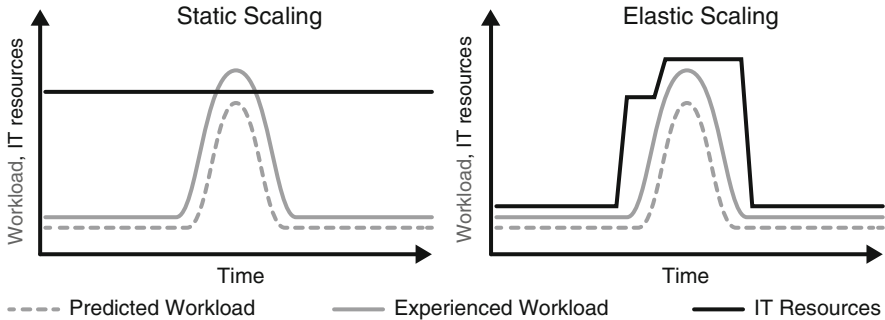


Fig. 2.5 Exemplary once-in-a-lifetime workload

decreases. The first increase is provisioned for the predicted workload. When the workload increases more than expected, the elasticity of the cloud enables the additional increase.

Result

Once-in-a-lifetime workload may be handled with the same automated mechanisms as *periodic workload* (29). In contrast to these automated mechanisms, the provisioning and decommissioning tasks necessary for *once-in-a-lifetime workload* may also be handled manually using the self-service capability of the used cloud offerings. As provisioning and decommissioning is only performed once, the benefits of an automated alignment of IT resource numbers to the experienced workload are reduced possibly making the additional effort to automate them unreasonable. Therefore, the provisioning of required IT resources, their integration into a company's IT landscape, possibly the assignment of workload to them, and their decommissioning may be handled by humans in these situations.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): this cloud service model enables the provisioning of many additional servers to be used by the *once-in-a-lifetime workload*. The integration of these servers into the customer's application, communication network etc. can often be realized as a manual task. Sometimes, the additional servers do not have to be integrated at all, but the workload assignment, it's processing, and the use of results is completely handled by users. This can be the case, if an employee needs one-time access to a powerful server provided by the *IaaS* offering.
- *Platform as a Service (PaaS)* (49): applications hosted on a provider-supplied *execution environment* (104) provided by a *PaaS* offering are often scaled automatically. However, in scope of *once-in-a-lifetime workloads*, the provider-supplied management functionality for elastic scaling may not be

reactive enough to detect and handle the extreme increase of resource demand automatically. Its behavior and configurability, therefore, has to be evaluated carefully in advance.

- *Software as a Service (SaaS)* (55): complete applications provided by *SaaS* offerings are commonly billed per month and users may sign up at any time. Therefore, a planned increase of application users to handle once-in-a-lifetime workload can usually be handled.
- *Public cloud* (62) and *community cloud* (71): all cloud deployment models are suitable for the handling of *once-in-a-lifetime workload*. The public and community cloud option is most flexible regarding the use of their resources. For workload peaks occurring only once in a lifetime, these deployment models, therefore, should be preferred.
- *Private cloud* (66): a private cloud can be less flexible as other cloud deployment models and establishing a private cloud often involves handling physical IT resources. Therefore, a *private cloud* is commonly only suitable for *once-in-a-lifetime workloads*, if the *private cloud* exceeds a critical size, thus, many applications of a company are hosted by it and experience changing workload. Problems may arise if *once-in-a-lifetime workload* peaks do occur for many of the hosted applications simultaneously.
- *Elastic infrastructure* (87) and *elastic platform* (91): these two patterns describe offerings providing virtual servers and execution environments for custom application components. Especially, they provide humans with self-service interfaces through which the resources necessary for the handling of *once-in-a-lifetime workload* may be provisioned and decommissioned.

Known Use

A good example for *once-in-a-lifetime workload* and one of the first well-known uses of cloud offerings is the digitalization of the New York Time archives. Four terabyte of digital scans of printed articles of the New York times between the years 1851 and 1980 were converted to Acrobat PDF documents using 100 virtual servers [31] dynamically obtained from Amazons Elastic Compute Cloud (EC2) [18]. The process was repeated and extended to provide full-text search by using an optical character recognition (OCR) tool [32]. The resulting article archive is available online and called the Time Machine of the New York Times [33].

2.2.4 Unpredictable Workload

IT resources with a random and unforeseeable utilization over time experience unpredictable workload.



How can random and unforeseeable utilization be characterized and how can applications experiencing this workload benefit from cloud computing?

Context

Random workloads are a generalization of *periodic workloads* (29) as they require elasticity but are not predictable. Such workloads occur quite often in the real world. For example, sudden increases of Website accesses due to weather phenomena or shopping-sprees when new products gain an unforeseen attention and public interest. The resulting occurrence of peaks or at least their height and duration often cannot be foreseen in advance under these conditions.

Solution

Unpredictable workloads require the unplanned provisioning and decommissioning of IT resources hosting applications. The necessary provisioning and decommissioning of IT resources is, therefore, automated to align the resource numbers to changing workload quickly and directly when it is being monitored. *Unpredictable workloads* are extremely hard to handle with static scaling shown at the left of Fig. 2.6. As the maximum workload peaks and the time when they occur are unknown, IT resources are often provisioned to a certain level that is economically feasible. If the workload exceeds what can be handled by these IT resources, the performance of the application degrades. Elastic scaling seen on the right of Fig. 2.6 can instead be used to monitor the experienced workload and base provisioning and decommissioning of IT resources on this information without relying on workload predictions. However, this requires a quick reaction to the workload change and very steep workload inclines can still be problematic as resources require time to be provisioned. If such strong workload inclines are anticipated, IT resources may have to be kept on standby, as described in greater detail by the *standby pooling process* (279) management pattern in Chap. 5.

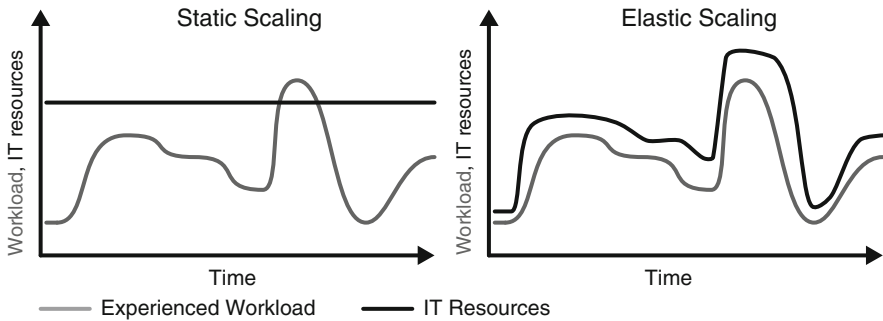


Fig. 2.6 Exemplary unpredictable workload

Result

As with *periodic workload* (29), providers have to be able to dynamically add and remove resources to customers during peak-workload times and remove them when workload intensity is lower. If one customer decommissions IT resources, the provider has to be able to assign these resources to another customer experiencing increased workload to offer a sustainable pay-per-use pricing model. Sometimes, providers move this problem partly to the customer side by offering resources for a reduced price if they are provisioned for longer time frames.

To benefit from the pay-per-use based pricing model, the customer has to be able to dynamically provision and decommission IT resources as well. As the peaks are random and unpredictable, utilization of resources is monitored and resource numbers are adjusted based on this information. The customer can either do this in custom developed management functionality or through configuration of provider-supplied management functionality. In either case, the application has to support the integration of newly provisioned IT resources as well as the removal of unused IT resources. Therefore, it has to be elastic itself as described by the cloud application property covered on Page 4 in Sect. 1.1.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): a cloud offering servers as *IaaS* commonly provides monitoring functions to detect workload increases. However, how to react to such situations often has to be configured or implemented by the customer. In this scope, it is very important to evaluate how long a server required by the application takes to be provisioned. In scope of *unpredictable workload*, the intensity of an increase may be very high creating the need to keep additional servers on standby as described by the *standby pooling process* (279) pattern.
- *Platform as a Service (PaaS)* (49): similar to the use of a provider-supplied *execution environment* (104) for custom applications experiencing *periodic workload* (29), the provider may supply management functionality handling

elastic scaling that has to be configured. Especially, the intensity of provisioning and decommissioning has to be specified directly affecting how well the application handles unpredictable peaks.

- *Software as a Service (SaaS)* (55): if complete applications are obtained as *SaaS* from a cloud offering, the workload manifests in the application being accessed by more users. As billing models are often based on monthly subscriptions, frequent workload peaks of *unpredictable workload* may hinder such offerings from being used efficiently. Nevertheless, customers still benefit from management tasks handled by the provider.
- *Public cloud* (62) and *community cloud* (71): these cloud deployment models are very flexible regarding the dynamic provisioning and decommissioning of resources and are, therefore, ideal for the handling of *unpredictable workload*.
- *Private cloud* (66): if a *private cloud* hosts few applications experiencing similar workload, simultaneous peaks of many handled applications can impose the problem of non-availability of IT resources. Therefore, the user-base must be large and diverse enough, so that workload peaks of applications can be leveled out regarding the overall utilization of the cloud.
- *Elastic infrastructure* (87) and *elastic platform* (91): to automate the provisioning and decommissioning of cloud resources, the cloud provider has to enable the customer to perform these tasks in custom applications and, thus, to monitor the utilization of an application to make good scaling decisions. The necessary provider interfaces are conceptually described by the *elastic infrastructure* (87), providing virtual servers as *Infrastructure as a Service (IaaS)* (45), and the *elastic platform* (91) hosting custom developed application components as *Platform as a Service (PaaS)* (49).
- *Elasticity manager* (250), *elastic load balancer* (254), and *elastic queue* (257): making adequate decisions when to provision and when to decommission IT resources is a challenging task. Applications and their runtime environment have to be monitored to identify overprovisioning or underprovisioning. These three patterns describe how the utilization of application components, requests assigned to them, or the number of messages exchanged with them can be used to make this decision, respectively.
- *Feature flag management process* (271) and *standby pooling process* (279): if the time it takes a cloud provider to provision new IT resources is too long for the anticipated workload increases, IT resources may have to be kept on standby. The *standby pooling process* (279) pattern describes how this can be handled. Additionally, feature flags may be used to degrade an application gracefully by keeping important functionality operational in case a sufficient amount of resources can still not be provisioned in time as described by the *standby pooling process* pattern.

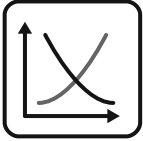
Known Use

One example for *unpredictable workload* is the workload that occurs when building systems for connected vehicles. Some traffic patterns, such as rush-hour or traffic

jams that concentrate workload on certain parts of the connected vehicle system can be anticipated. However, accidents and weather conditions, that require spontaneous re-routing requests or produce traffic jams that result in a higher load of requests to the central servers, cannot be predicted up front.

2.2.5 Continuously Changing Workload

IT resources with a utilization that grows or shrinks constantly over time experience continuously changing workload.



How can a continuous growth or decline in utilization be characterized and how can applications experiencing this workload benefit from cloud computing?

Context

Many applications experience a long term change in workload. Increasing workload often corresponds to the successful growth of a business after it was launched impacting the supporting applications. Decreasing workload is often experienced by legacy applications that still handle some of the workload that is slowly fading or by applications supporting discontinued products that are continuously used by fewer customers. In both cases – growing and shrinking workload – IT resources need to be provisioned or decommissioned with varying intensity. This growing and shrinking can be planned or unplanned, i.e., it can be previously known at which rate the growth or shrinking takes place or not.

Solution

Continuously changing workload is characterized by an ongoing continuous growth or decline of the utilization. This change can be linear, non-linear, exponential etc. but in any case, the change in utilization is consistent towards one direction. Elasticity of clouds enables applications experiencing *continuously changing workload* to provision or decommission resources with the same rate as the workload changes. Figure 2.7 shows an exemplary consistently increasing workload. The depicted static scaling and elastic scaling may be used analogous for continuously decreasing workload. In case of static scaling seen at the left of Fig. 2.7, IT resources are provisioned stepwise with the increasing workload. These large increments are present, because physical hardware, such as servers, are more efficiently provisioned in large bulks as manual tasks are involved. Elastic scaling on the other hand align the resource increments tightly to the increasing workload, because IT resources may be provisioned more flexibly and one by one as seen on the right of Fig. 2.7.

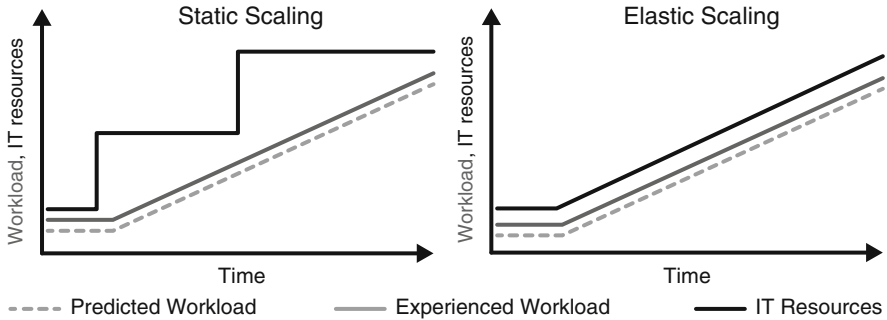


Fig. 2.7 Exemplary continuously changing workload

Result

If the rate of workload change is known and not very intense, the same effects apply as with planned *once-in-a-lifetime workload* (33). IT resource provisioning and decommissioning can be performed partially as manual activity via the self-service interface of the cloud offering. For a provider of a cloud or within a *private cloud* (66), a combination of applications experiencing increasing workload and decreasing workload can be used to level out the modification rate of IT resources. This approach is especially promising if a legacy application is replaced with a new application and both applications share a cloud, because the combined workload can be *static workload* (26).

In cases of *continuously changing workload* where the rate of workload change is varying or unknown, the same challenges arise as with the *unpredictable workload* (36), because the elastic scaling has to be adjusted automatically to the rate at which growing or shrinking takes place.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): pricing models used by *IaaS* cloud providers offering virtual servers commonly support *continuously changing workload* very well. Servers used by an application can be provisioned with increasing workload or decommissioned with declining workload. At a certain point, when an application is not used anymore at all, a *IaaS* provider often supports that virtual servers are shut down and persisted enabling the customer to restart and application on demand if it is needed again later.
- *Platform as a Service (PaaS)* (49): a provider-supplied *execution environment* (104) offered as *PaaS* behaves similar to an *IaaS* offering when facing *continuously changing workload*. As scaling may be handled by the provider, IT resources are provisioned and decommissioned automatically with increasing or declining workload, respectively. The intensity of these operations may have to be configured by the customer to match the continuously changing workload in a

concrete usage scenario. Similar to an *IaaS* offering, *PaaS* offerings often allow the suspend of an application at which point only minimal costs are generated.

- *Software as a Service (SaaS)* (55): if complete applications provided by the cloud as part of a *SaaS* offering are billed per user, use of such an offering to handle *continuously changing workload* is only beneficial if the number of users actually changes. If the same number of users access the application during a workload increase or decline – only more or less often – the costs generated by the application remain constant.
- *Public cloud* (62) and *community cloud* (71): *continuously changing workload* is especially suitable for these environments as they are less restricted regarding the number of IT resources that can be provided by them.
- *Elastic infrastructure* (87) and *elastic platforms* (91): just as for *periodic workload* (29) and *once-in-a-lifetime workload* (33), the self-service interfaces of these cloud offerings can be used to automate the provisioning and decommissioning of IT resources as the workload changes providing the required elasticity.
- *Elasticity manager* (250), *elastic load balancer* (254), and *elastic queue* (257): these three patterns describe how the change in workload may be monitored to determine the necessary adjustments in resource numbers.

Known Uses

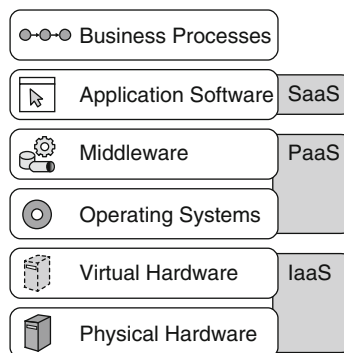
An example for a constant decline in utilization is experienced by custom applications of manufacturers supporting specific products, i.e., provide customer manuals, service information etc. A product that is no longer in production will eventually vanish from the market, thus, continuously reducing the workload experienced by the application accessed, for example, by customers, retailers, or technicians.

2.3 Cloud Service Models

Just as the NIST cloud definition [3], the following patterns compare and categorize different cloud service models according to the layers of the application stack for which they provide IT resources. Figure 2.8 shows the application stack that we use throughout the book to illustrate on which layer certain cloud offering resides. The layers map to the NIST definition of cloud service models: *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)* and *Software as a Service (SaaS)*. From bottom to top the six layers comprising the stack are:

- **Physical hardware:** tangible physical infrastructure. This infrastructure contains, for example, servers, storage, networks connecting servers and racks containing the servers, as well as the building housing the data center, power lines etc.
- **Virtual hardware:** physical hardware components can be abstracted and mapped to virtual counterparts by a *hypervisor* (101) and *virtual networking* (132). The aim of this mapping is to share physical hardware between multiple

Fig. 2.8 Application stack and associated cloud service models



virtual counterparts, for example, virtual servers are mapped to fewer physical servers. This ensures that users of the virtual servers perceive the system as if they were the only one accessing it while physical hardware can be shared. Furthermore, virtualization reduces the need to physically adjust tangible resources, such as servers, networking switches, cables etc. in a datacenter when configurations have to be changed.

- **Operating system:** software installed directly on the physical or virtual hardware. Operating systems abstract hardware by providing functions to applications installed on the operating system, for example, to access network cards or files stored on the hard drive. Examples for operating systems on this layer are Microsoft Windows Server, Linux, or Apple OS X Server.
- **Middleware:** software on this layer is installed on an operating system and itself provides an environment for installation and execution of custom applications, processes, and data. Examples for such environments range from execution of certain programming languages, such as Python or the Java Virtual Machine to more complex middleware products hosting custom applications later described by the *execution environment* (104) pattern. Examples for more complex middleware are application servers such as JBoss or IBM Websphere, workflow engines such as Apache ODE, or IBM Websphere Process Server. Middleware can also provide communication services later characterized by the *message-oriented middleware* (136) pattern, for example, messaging by Apache ActiveMQ or IBM Websphere MQ. Data storage can also be handled by middleware, for example, MySQL, Oracle 11g, or IBM DB2. Such functionality is described by the storage offering patterns in Sect. 3.5 on Page 109.
- **Application software:** custom applications providing functionality to human users or other applications are associated with this layer. Examples for applications are software for customer relationship management systems (CRM) or enterprise resource planning (ERP).
- **Business processes:** the processes of a company that are supported by a set of applications are associated with this layer. These processes are domain specific and subsume, for example, order processing, credit approval processes, billing etc.

Cloud offerings can reside on any of these layers. *Infrastructure as a Service (IaaS)* (45) makes (virtual) hardware accessible to customers. This can be servers, network resources or storage. *Platform as a Service (PaaS)* (49) maintains an *execution environment* (104) for customers to deploy individual applications or components thereof. *Software as a Service (SaaS)* (55) offers a complete application that is accessible by humans through a graphical user interface or within the implementation of custom developed applications using an application programming interface (API).

The workload patterns that we covered in the previous section can occur on any level of the stack – be it on infrastructure, platform, software, or business process level. The level of interest for a customer is the level at which he or she outsources workload to the cloud. Thus, when outsourcing the infrastructure level, the workload patterns at the infrastructure level, i.e., server utilization and network load are of interest. Similar, the workload patterns at the software level, for example, user requests or the number of application users determine the necessity for elasticity when outsourcing on the *SaaS* level.

In the following, we cover each of the cloud service models in more detail, describe which part of the application stack is provider-supplied and which part is handled by customers, and highlight the specific features and properties with regard to the essential cloud properties: access via network, on-demand self-service, measured service (pay-per-use), resource pooling, and rapid elasticity. Also, we point to cloud offering patterns covered in Chap. 3 that describe the functionality provided on the different levels of the application stack in greater detail.

2.3.1 Infrastructure as a Service (IaaS)

Providers share physical and virtual hardware IT resources between customers to enable self-service, rapid elasticity, and pay-per-use pricing.



How can different customers share a physical hosting environment so that it can be used on-demand with a pay-per-use pricing model?

Context

Applications often experience varying workloads that lead to different utilizations of IT resources on which these applications are hosted. Especially, in the scope of *periodic workloads* (29) with reoccurring peaks and the special case of *once-in-a-lifetime workloads* (33) with one dramatic increase in workload, IT resources have to be provisioned flexibly. One common hosting environment for applications is formed by independent servers on which applications are installed. With changing workload, the number of these servers shall be adjusted.

Solution

A provider using the *Infrastructure as a Service (IaaS)* service model offers physical and virtual hardware, such as servers, storage and networking infrastructure that can be provisioned and decommissioned quickly through a self-service interface. On these IT resources, customers install their individual operating systems, middleware, and applications software supporting their business processes as depicted in Fig. 2.9.

Result

IaaS clouds in detail offer infrastructure IT resources such as servers, storage (volatile memory and persistent disk storage) and networking. Many *IaaS* providers offer customers the choice to provision resources to multiple data center locations, allowing the distribution of applications and platforms across multiple geographic locations. How an *IaaS* cloud behaves is covered in detail by the cloud offerings patterns in Chap. 3. In this scope, the *elastic infrastructure* (87) pattern describes a common combination of cloud offering to provide *IaaS* to customers (see the

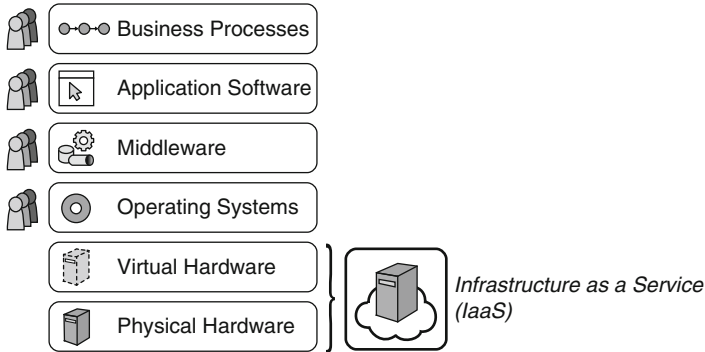


Fig. 2.9 Infrastructure as a service in the application stack

related patterns section for a detailed list). The cloud computing properties introduced in Sect. 1.1 on Page 3 are enabled in an *IaaS* cloud as follows.

Access via network: remote access to provided servers and to the storage disks is one of the key features of *IaaS*. Typically, after starting a virtual server it can be accessed remotely via a secure shell (SSH) or graphically, for example, using the remote desktop protocol (RDP) depending on the operating system running on the server and its configuration. Customers typically get full access to the servers.

On-demand self-service: customers can often access *IaaS* offerings via a Web-portal that allows customers to configure and provision servers, storage and network connectivity. One important aspect of such an on-demand self-service portal is the monitoring which allows supervising the status of the provisioned IT resources, their configuration and the corresponding charges. This information can also often be extracted from the cloud provider in an automated fashion using an application programming interface (API).

Pay-per-use: pricing models for *IaaS* are often based on an hourly charge for servers, amount of data stored per month, and amount of data exchanged via the cloud providers network per month. Prices for servers range from a few cents per hour for small servers to multiple dollars per hour for larger possibly clustered servers. To ease the calculation and comparison with traditional server processing power, volatile memory and disk storage costs are commonly determined according to a certain set of server configurations that resemble traditional servers. Often, several sizes of these server configurations are available resembling T-shirt sizes such as S, M, L, XL, XXL etc. Regarding the data exchanged with a cloud, different prices are often allocated for data uploaded to the cloud and data downloaded from the cloud. Therefore, when storing data in a cloud, the costs for transferring it and the additional monthly costs for storing it in the cloud have to be considered.

Resource pooling: resources of an *IaaS* cloud are shared between customers on the hardware infrastructure level. Thus, the IT resources pooled between different

customers of an *IaaS* cloud typically are: the physical data center, physical servers, physical storage disks, and physical network components such as routers, switches, cables and firewalls as well as the personnel to maintain and operate the data centers.

Rapid elasticity: the flexible use of servers is a key feature of *IaaS* clouds leading to their success and differentiating them significantly from other server hosting offerings. In a typical *IaaS* offering, new servers are started within minutes, firewalls and storage disks are provisioned almost instantly. Similarly all billed resources can be returned to avoid further charging, within minutes. This rapid elasticity on the infrastructure level enables adding and removing infrastructure resources on-demand.

Traditional server hosting could be compared to *IaaS*. However, these offerings often lack the dynamic pay-per-use billing model and rapid elasticity. Instead users pay monthly fees for servers whether they use them or not. Also, the ability to provision and decommission virtual servers via a self-service interface that can be accessed programmatically is often unavailable. However, the billing models of these hosting providers are sometimes also available at *IaaS* providers. Users may then decide between pay-per-use billing and a lower fixed price for resources that are provisioned for longer time periods.

Related Patterns

- *Elastic infrastructure* (87): an *IaaS* offering can subsume multiple services and functionality. A typical collection of this functionality is covered in greater detail by the *elastic infrastructure* pattern. The covered functions subsume repositories in which templates for servers including pre-installed and configured operating systems, as well as selected applications components or middleware are managed. Commonly, these images can be maintained by the cloud provider or can be created by the customer or a user community. Furthermore, an *elastic infrastructure* typically offers an application programming interface (API) through which its functionality can be accessed. Thus, it becomes possible to dynamically provision and decommission (virtual) machines, storage and network elements. APIs are of particular importance when building more sophisticated *Platform as a Service (PaaS)* (49) offerings and *Software as a Service (SaaS)* (55) offerings that use these APIs to automatically scale on demand. Some of the functionality subsumed by an *elastic infrastructure* can also be offered as an independent cloud offering. Therefore, these offerings have been described as the separate patterns *hypervisor* (101), *block storage* (110), and *virtual networking* (132).
- *Hypervisor* (101): many cloud providers use hardware virtualization to host multiple servers provided to customers on an *IaaS* basis on a single physical server. The basic concepts of this virtualization are described by the *hypervisor* pattern. In almost all cases of publically available *IaaS* clouds, thus, *IaaS* offerings using the *public cloud* (62) deployment model, virtualization is used

as the basic implementation to realize the virtual server configurations. However, a few providers also offer physical servers as *IaaS* offering providing the same cloud properties described in Sect. 1.1 on Page 3.

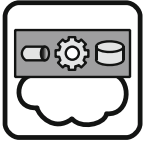
- *Block storage* (110): as mentioned above disk storage is often provided by an *IaaS* offering in addition to servers. This storage can be kept independent from virtual servers in case a server is shut down or fails. The behavior of such offerings providing disk storage is described by the *block storage* pattern.
- *Virtual networking* (132): one of the resources typically provided and billed in an *IaaS* offering is the connection network and data exchanged through it. The network aspect of *IaaS* does not only include the bandwidth but often also firewalls that can be configured to allow access to and from certain ports over certain protocols to the resources started in the *IaaS* cloud. The virtual networking pattern describes how this aspect of an *IaaS* offering behaves.

Known Uses

Since the advent *IaaS* offerings, this type of cloud offering has gained widespread acceptance. A large number of offerings in form of *public clouds* (62) and in the form of virtual *private clouds* (66) have been established. Amazon's Elastic Compute Cloud (EC2) [18] and Rackspace [19] are *public cloud* (62) providers providing an *IaaS* offering. Other companies, such as T-Systems with their Dynamic Services for Infrastructure (DSI) [34] provide *IaaS* clouds for virtual *private cloud* (66) scenarios. In addition to these cloud offerings, a wide range of tools and data center automation products have been established, both in the open source and commercial market that enable companies and providers to build their own *private clouds* (66) offering *IaaS*. VMware's vCloud [20] is an example for a commercial implementation. OpenStack [35] is an open source implementation that is also used and supported by Rackspace [19]. Similar open source software for the management of an *IaaS* clouds is OpenNebula [36] and Eucalyptus [37]. Each of these implementations support most of the functionality common for an *elastic infrastructure* (87).

2.3.2 Platform as a Service (PaaS)

Providers share IT resources providing an application hosting environment between customers to enable self-service, rapid elasticity, and pay-per-use pricing.



How can custom applications of the same customer or different customers share an execution environment so that it can be used on-demand with a pay-per-use pricing model?

Context

In Sect. 2.2, various workloads have been covered and how applications experiencing them can benefit from an elastic cloud. Especially, *periodic workloads* (29), *once-in-a-lifetime workloads* (33), and *unpredictable workloads* (36) have shown suitable for this, but other type of workloads could also benefit from a cloud environment. *Infrastructure as a Service – IaaS* (45) flexibly provides servers to customers of applications for this purpose. However, this means that customers have to install and manage their own operating systems, middleware, and *execution environments* (104), such as a Java Virtual Machine, an application server, webserver, databases etc. If many customers require similar hosting environments for their applications, there are many redundant installations resulting in an inefficient use of the overall cloud. Furthermore, the management complexity to maintain an operating systems and middleware has to be handled by each customer of the *IaaS* offerings. Especially, small and medium-sized businesses may not have the manpower and skills to perform these tasks efficiently and thoroughly.

Solution

A cloud provider using the *Platform as a Service (PaaS)* service model offers managed operating systems and middleware. Customers may host individual application software supporting their business processes in this environment as depicted in Fig. 2.10. Management, such as updating operating systems or middleware is handled by the provider. Often, the provider expects custom applications to be developed in a certain style to additionally handle the elastic scaling and failure resiliency of applications for the customer.

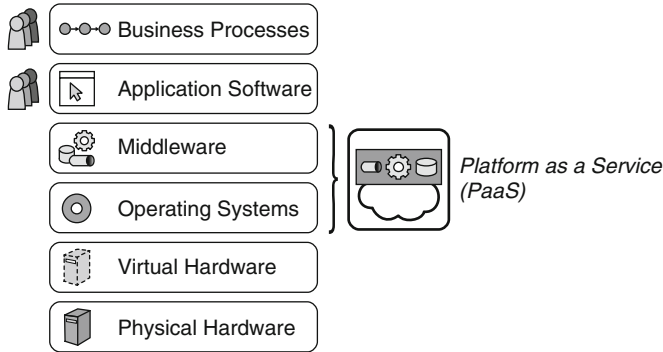


Fig. 2.10 Platform as a service in the application stack

Result

Platform as a Service (PaaS) subsumes the layers above physical and virtual hardware and below complete software applications – thus, it contains operating systems as well as middleware products, i.e., database management systems, application servers, *message-oriented middleware* (136) etc. How these individual services behave is covered in detail by the cloud offerings patterns in Chap. 3. In this scope, the *elastic platform* (91) pattern describes a common combination of these offering to provide *PaaS* to customers (see the related patterns section for a detailed list). The cloud computing properties described in Sect. 1.1 on Page 3 are enabled as follows in a *PaaS* offering.

Access via network: commonly, customers of a *PaaS* offering can access the offered *execution environments* (104) hosting an application via a network, most prevalent either an intranet or the Internet, to deploy their applications. Some *PaaS* clouds also offer development environments accessible for customers via a network to build and test their applications.

On-demand self-service: similar to an *IaaS* (45) cloud, *PaaS* clouds offer a self-service portal or an application programming interface (API) through which customers can deploy applications. In contrast to an *IaaS* cloud, the *PaaS* self-service interface often does not display virtual servers and does not allow starting and stopping individual servers. The *PaaS* self-service interface is instead focused on the offered environment that scales independently, thus, no scaling functionality is developed by the customer. The self-service portal also displays the consumed IT resources as well as their health status, i.e., their current availability and statistics for uptime and utilization.

Pay-per-use: as a *PaaS* offering provides an environment to host applications directly billing is performed based on the use of functionality provided by this environment. A *PaaS* provider can bill for different functionality, i.e., per user access to a hosted application, per message exchanged by that application etc. For example, Amazon bills per message queued in their Amazon Simple Queue

Service (SQS) [38]. Storage offerings are typically billed either per amount of items stored or per storage volume. Web application runtimes can be billed per amount of handled requests or per amount of exchanged data. *PaaS* offerings, thus, commonly bill according to the type of IT resources they offer, instead of billing for servers or network bandwidth, as *IaaS* (45) offerings do.

Resource pooling: when discussing *PaaS*, lines can be blurred between “native” *PaaS* offering and a managed *IaaS* offering with pre-installed and managed middleware components – a variation of *PaaS* covered in greater detail by the *elastic platform* (91) pattern. One important distinction is that a *PaaS* offering pools resources on the middleware layer whereas an *IaaS* offering pools resources on an infrastructure layer. Thus, in a *PaaS* storage offering, for example, multiple customers share the same middleware whereas they would have a distinguished instance of the middleware in an *IaaS* (45) cloud. However, to enable this sharing, isolation between customers has to be ensured. The consequence of sharing on the middleware layer is that the offered middleware has to be redesigned to be multi-tenant aware: for the customer it should appear that he or she is using a dedicated instance of the middleware. The provider can then balance load on the middleware layer, preventing the need to start and stop underlying virtual servers frequently. In a native *PaaS* isolation of tenants is also guaranteed through some throttling mechanisms, preventing that one customer uses up the resources assigned to the shared runtimes with faulty applications or by overloading the environment on purpose.

Rapid elasticity: flexible scaling on the platform layer requires the cloud provider to dynamically add and remove IT resources from individual customers depending on their demands. However, as *PaaS* clouds are characterized by their sharing capability on the middleware layer, this problem can be delegated to a load balancer. This load balancer distributes the requests of customers to the middleware instances that have the applications of the respective customer installed. Deployment of applications to middleware instances is handled in a transparent manner to the customer, either at deployment time or on-demand. Therefore, additional requests can be handled without modifying the applications. However, when all customers start to increase or decrease their number of requests, at some point more middleware instances need to be added to or removed from the *PaaS* cloud. This can either be done automatically via an underlying *IaaS* infrastructure or manually if the *PaaS* is not built upon an *IaaS* cloud. However, it is important that once the platform is scaled out with new middleware instances, these instances become able to serve a customer’s requests. Again this is ensured by automatically deploying the customer’s application on them.

Therefore, the key to be able to rapidly increase the amount of requests to be handled from one customer is the ability to dynamically and transparently distribute these requests among multiple instances of the middleware and, thus, among multiple (virtual) servers. This is similar to a traditional cluster with the addition of multi-tenancy, i.e. the ability to serve multiple customers concurrently from

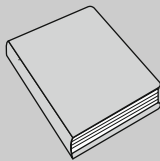
these middleware instances while guaranteeing isolation between those customers. In case of provider-managed scaling of the *PaaS* cloud, applications often have to be developed in a certain way to be deployed to multiple instances transparent to the user. Often, applications have to rely on external state information as described by the *stateless component* (171) pattern and by the *elastic platform* (91) pattern.

Related Patterns

- *Elastic platform* (91): a *PaaS* offering commonly consists of a certain set of functionality. As this functionality can be provided as individual cloud offerings, we described them as individual patterns. The *elastic platform* pattern describes how these other patterns are commonly combined to form a *PaaS* offering. The central functionality of a *PaaS* offering is an *execution environment* (104). Furthermore, the *elastic platform* can provide storage functionality in the form of a *blob storage* (112), *relational databases* (115), and *key-value storage* (119) as well as communication functionality to exchange information between hosted applications and external applications. This is covered in greater detail by the *message-oriented middleware* (136) pattern.
- *Execution environment* (104): this pattern describes how an environment can be provided in which customer-developed applications can be deployed and executed. Especially, it covers how functionality required by many applications can be provided by this environment so that implementation complexity is reduced for the customer.
- *Blob storage* (112), *relational database* (115), and *key-value storage* (119): an *elastic platform* (91) can provide file-system-like *blob storage*. Table-centric storage that supports more complex querying functionality that the retrieval of files is provided by *relational databases* and *key-value storage*.
- *Message-oriented middleware* (136): communication in cloud applications should be asynchronous to enable the loose coupling cloud application property described on Page 7 in Sect. 1.2. We cover how this property can be enabled in greater detail in the *loose coupling* (156) pattern as well.
- Cloud application architecture patterns (Chap. 4): applications deployed on an *elastic platform* (91) offered as *PaaS* should follow certain architectural principles to benefit efficiently from this environment. The patterns in Chap. 4 describe these principles as patterns. Most importantly, an application running on an *elastic platform* often has to be distributed, as described by the *distributed application* (160) pattern. Furthermore, it is important to consider where the application holds its state. Often, *PaaS* providers recommend or enforce that their own storage offerings mentioned above are used for this purpose, thus, the application must be implemented using *stateless components* (171). When interacting with the communication functionality, it is important to consider the delivery behavior. Messages can be delivered *at-least-once* (144) or *exactly-once* (141) using a *transaction-based delivery* (146) or *timeout-based delivery* (149) protocol. The application components interacting with this provider-supplied

functionality, therefore, may need to deal with message duplicates by implementing the *idempotent processor* (197) pattern. Furthermore, they may need to extend the delivery assurance of messages to assure their successful processing, by implementing the *transaction-based processor* (201) pattern or the *timeout-based message processor* (204) pattern. When interacting with cloud storage offerings provided as *PaaS*, data consistency assured by the provider also has to be considered as described by the *strict consistency* (123) and *eventual consistency* (126) patterns.

- Cloud application management patterns (Chap. 5): some *PaaS* providers handle the elastic scaling of hosted applications for the customer. If this task, however, has to be handled by the customer himself or herself, the cloud application may need additional management components (see Sect. 5.2 on Page 242) that monitor the workload experienced by the application and provision or decommission resources accordingly. An *elasticity manager* (250) does this based on the utilization information obtained about application components. An *elastic load balancer* (254) monitors the number of synchronous requests to the applications and an *elastic queue* (257) the number of asynchronous messages. The behavior implemented by these management components is covered in greater detail by management process patterns starting in Sect. 5.3 on Page 264.



Further Reading: there are additional patterns that cover application design. Even though most of these patterns did not have *PaaS* in mind when they were written, many of them are applicable to applications deployed on a *PaaS* offering. Hope and Woolf [1] cover patterns for message-based enterprise application integration. These patterns are also used by pattern of this book and are summarized by *the message-oriented middleware pattern* (136). Buschmann et al. [14], Gamma et al. [2], and Fowler [15] also describe patterns that can be considered when designing application components. Buschman et al. describe architectural patterns applicable to standalone and distributed applications. Gamma et al. cover best practices for object-oriented programming. Fowler's enterprise architecture patterns describes, for example, how a business usage scenario and the data and functionality it relies on may be mapped to data structures and application functions to be implemented.

Known Uses

PaaS offerings, such as Google's App Engine [21] or WSO2's Stratos Live [39] allow the deployment of (Java) Web applications. Others offer an *elastic platform* (91) to deploy and create business processes, for example, Metasonic [40], RunMyProcess [41] or Cordys [42]. In the virtual *private cloud* (66), offerings such as T-System's Dynamic Services for SAP solutions [43] also exist. Another well-known *PaaS* offering is Force platform [44] of salesforce.com which allows consumers to build extensions to the Salesforce CRM *SaaS* offering [45]. The Amazon Simple Queue Service (SQS) [38] offers *message-oriented middleware* (136) in the cloud. Other *PaaS* offerings do not only provide an *elastic platform* (91), but also additional services required in a certain business domain, for example, data about the stock market or systems to build and test customer-developed applications. An example of the latter is CloudBees [46] which offers a complete environment for software development and hosting.

2.3.3 Software as a Service (SaaS)

Providers share IT resources providing human-usable application software between customers to enable self-service, rapid elasticity, and pay-per-use pricing.



How can customers share a provider-supplied software application so that it can be used on-demand with a pay-per-use pricing model?

Context

The workload patterns covered in Sect. 2.2 have shown that different workloads experienced by an application make them more or less suitable for an elastic cloud environment. Especially, workload peaks experienced in scope of *periodic workload* (29), *once-in-a-lifetime workload* (33), and *unpredictable workload* (36) can be handled efficiently by a cloud environment. However, a prerequisite for this decision is that the workload of applications has to be measured. Then, a cloud offering to host that application has to be selected. Regardless whether this offering follows the *IaaS* (45) or *PaaS* (49) service model, the application needs to be developed and deployed. Especially, small and medium enterprises may not have the manpower and know-how to develop custom software applications for this purpose. Other applications have become commodity and are used by many companies, for example, office suites, collaboration software, or communications software. However, if these applications are shipped to customers to be installed on IT resources managed by customers, the complexity to provision and maintain these resources or to select a suitable cloud provider still has to be handled by the customer.

Solution

A provider using the *Software as a Service (SaaS)* service model offers a complete software application to customers who may use it on-demand via a self-service interface. The provider, therefore, provides customers with application software to support their individual business processes as depicted in Fig. 2.11. Customers perform their individual business processes, but do not have to install and manage an application required to support these processes. Accesses to this application are billed on a pay-per-use basis.

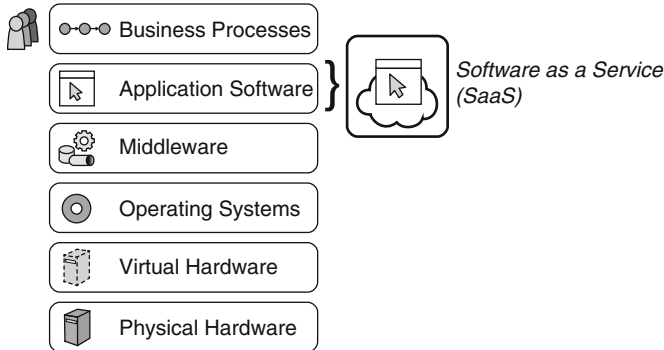


Fig. 2.11 Software as a service in the application stack

Result

SaaS is the established cloud service model in which the advantages of the cloud principles become most obvious. Instead of buying and installing hardware, paying for licenses, handling installation and configuration of the necessary middleware and software products, training and paying for system administrators on the hardware, middleware and software level, customers can simply obtain the required software from the cloud. In general, all applications could be provided by a *SaaS* cloud. Typical *SaaS* clouds provide application software for customer relationship management (CRM), collaboration, video conferencing and document management. Common to all *SaaS* offerings is the fact that a certain domain-specific functionality is offered, along with service level agreements (SLA) that guarantee the availability of that functionality. Applications offered as *SaaS* are managed, updated, and maintained by the provider. Customers can start using the provided applications in minutes instead of spending a significant amount of time to setup software applications on their own premises. Often, *SaaS* clouds allow the individual customers to configure and customize the application within defined boundaries, ranging from the specification of the user interface colors and logos to the configuration of workflows and custom data fields. The essential cloud computing properties introduced in Sect. 1.1 on Page 3 are enabled by a *SaaS* cloud as follows.

Access via network: in most cases, *SaaS* clouds provide Web-applications that offer a user interface accessed via HTTP over the Internet or an intranet of a company. Additionally, the functionality provided by the application can often also be accessed remotely via an application programming interface (API) to be integrated with other custom applications of a customer.

On-demand self-service: the Web-based user interface of a *SaaS* application provides management functionality through which customers can sign up for access to a *SaaS* offering. In this management user interface a customer can often test and evaluate the application(s) in question and can book the application. The customer will then be given access to the respective application and is

also commonly provided with information about the application status, bills etc. The management interface also allows the customer to issue trouble tickets if needed. The self-service portal enables customers to add and remove additional users for the application, configure the application and provide access to the application's API in order to being able to integrate it with other applications of the customer.

Pay-per-use: the realization of pay-per-use depends on the domain of the *SaaS* application. Strictly speaking the unit of payment should be dependent on the objects this application deals with. For example, in a Web conferencing system, pay-per-use could mean that the provider charges for every minute a user accesses a Web-conference. This corresponds to the load that is put on the offering, in case customers conduct conferences, the software generates load and, thus, the provider bills the customer. Typically, for *SaaS* offerings where the load is not tied directly to an object managed by the system, or where the computation would be very complex, schemes that require a payment per user per month are established. It is assumed that each user that uses the application will put load on the system and, thus, costs occur for the provider.

Resource pooling: sharing of IT resources of a *SaaS* offering occurs on the software level. As a result different customers share not only the hardware and infrastructure resources as in the *PaaS* model but also the software resources. In this scope, customers are also called tenants, i.e., companies that have multiple employees accessing the application on their behalf. To prevent that different tenants interfere with each other, the application offered as a *SaaS* application must be multi-tenant aware and, thus, isolate tenants from each other, especially regarding the individual tenant's data. The advantage of multi-tenancy on the application layer is that the underlying middleware and infrastructure do not necessarily need to be multi-tenant aware as the isolation is handled on the application level. As an alternative to a multi-tenant solution, each tenant may be assigned a complete application stack for itself. This type of deployment is used by application service providers (ASP). However, the static fashion by which resources are assigned to tenants in this model hinders providers to share resources and scale elastically.

Rapid elasticity: in the context of a *SaaS* offering, rapid elasticity means that the offered software supports customers with a *periodic workload* (29), *unpredictable workload* (36), or *continuously changing workload* (40) profile, regarding the offered functionality. For example, for the teleconferencing offering this means that customers can book none or multiple teleconferences at once without necessarily having to pre-book them in advance or paying for one teleconference all the time even when they do not use it. Thus, as a consequence, the provider must balance the workload of the different tenants within the application. If this balancing is not possible an *elastic platform* (91) or *elastic infrastructure* (87) should underpin the application to be able to add and remove resources dynamically, if needed.

Related Patterns

The patterns in this book describe how cloud applications can be built on top of *IaaS* (45) or *PaaS* (49) offerings. The cloud application properties described in Sect. 1.2 on Page 5 are enabled through the use of cloud application architecture patterns of Chap. 4 and the cloud application management patterns of Chap. 5. Especially, the cloud application properties also make applications suitable to be offered as *SaaS* themselves. The following patterns are especially relevant to meet the challenges arising when offering applications to multiple tenants as a service:

- *Distributed application* (160): this pattern describes how an application's functionality may be decomposed to be distributed among several IT resources. This separation makes the application easier to scale elastically to handle the workload of all customers. It also assures that customers may configure on a finer granular which parts of the application shall be shared with other customers.
- *Stateful component* (168) and *stateless component* (171): an important aspect to consider in shared *SaaS* applications is the storage of data as customers are very sensitive to it. Therefore, application components that do not hold any state information are much easier to share between customers.
- *Data access component* (188): even though customers often require similar functionality, for example, for collaboration, schedules, project management etc. the data handled by customers may be domain specific. While every customer may, for example, manage his or her own customers using a *SaaS* application, the data associated stored for each customer is likely to be similar to other users of the *SaaS* application but may require some additional information. For example, a clothing retailer may want to store body size information for each customer in addition to common information, such as shipping addresses and billing information. In order to fulfill the requirements of all customers, a *SaaS* application, therefore, often has to be configurable regarding the data format it supports. How this can be achieved in a cloud application is covered by the *data access component* pattern.
- Multi-tenancy patterns (starting in Sect. 4.4 on Page 208): these patterns describe how application components comprising a *SaaS* application can be shared between different customers. Three different levels of sharing are covered in this section. *Shared components* (210) provide the same functionality to all tenants and do not support isolation. *Tenant-isolated components* (214) allow the tenant-specific configuration of the provided functionality and also ensure isolation between tenants. *Dedicated components* (218) are a portion of the application that is provided exclusively for a tenant. Often, *SaaS* (55) providers offer different implementations of application components and let the customer decide to which degree application functionality shall be shared with others.



Side Note: the patterns *elastic infrastructure* (87) and *elastic platform* (91) describe what kind of behavior users of *IaaS* (45) clouds and *PaaS* (49) clouds may expect, respectively. To match the cloud service model descriptions (Sect. 2.3), one could expect patterns describing the same aspects for *SaaS* (55) clouds. However, these are missing for two reasons. First, *SaaS* clouds differ significantly in behavior and functionality, depending on the business domain they support. Therefore, in these clouds, there is no common behavior by the time of this writing that could be abstracted to a pattern. The second reason, why these patterns are missing is that *SaaS* clouds do not form the basis for applications whose architecture can be based on the patterns described in this book. In scope of *SaaS* no custom code is deployed whose architecture would have to be considered. Some applications offered as *SaaS* can be extended using custom code in which case, the *SaaS* provider commonly offers a well-integrated *PaaS* cloud to host custom extensions to the *SaaS* application.

Known Uses

Salesforce.com offers a Web-based customer relationship management (CRM) software [45] as a *SaaS* offering. As customers demand extensibility and configurability of this application, salesforce.com also offers the Force platform [44] an *elastic platform* (91) as *PaaS* (49) to host such custom extensions, i.e., to integrate the *SaaS* CRM software with customers' other, possibly on-premise applications. Microsoft offers its complete office and collaboration suite as a Service, as part of Office 365 [47]. IBM offers collaboration software as part of its IBM SmartCloud [48]. Similar products, subsuming office and collaboration functionality, are available from Google Apps [49] as well. Telco providers, such as Deutsche Telekom started to offer *SaaS* applications, for example via their Business Marketplace [50] offering which allows independent software vendors (ISVs) to provide their applications in a *SaaS* model, where the telco provider handles automatic provisioning, billing, and support.

2.4 Cloud Deployment Models

Regardless of the service model followed by a cloud provider, a cloud can be hosted in different forms. These cloud deployment models can be mainly differentiated by the user groups accessing a cloud and the degree by which IT resources hosting the cloud itself are shared between customers. Regarding the accessibility of a cloud, the following cloud deployment types are introduced by the NIST cloud definition [3]: *public clouds* – generally accessible to everyone, *private clouds* – accessible only by a single institution, *community clouds* – accessible to a controlled group of institutions, and *hybrid clouds* – combining any set of other clouds. The NIST has further identified a number of deployment scenarios regarding the physical resources on which the different cloud deployment models may be hosted [51].

The following patterns cover these cloud deployment models and different ways to host them in detail. In this scope, we use the term “tenant” for companies or individuals that act as a customer of a cloud. Each tenant may have multiple users associated with it, i.e., a company may act as the customer of a cloud that is then used by employees of that company. We characterize each cloud deployment model regarding the number of tenants accessing the cloud and the number of tenants sharing IT resources hosting the cloud itself. For example, access to a cloud may be restricted to a certain user group, while the IT resources hosting the cloud itself may still be shared with others. Furthermore, we investigate what the different restrictions in accessibility mean for customers and providers and how the different cloud properties (see Sect. 1.1 on Page 3) access via network, on-demand self-service, measured service (pay-per-use), resource pooling and rapid elasticity influence the use of different cloud deployment models.

The restriction of the number of tenants accessing a cloud and sharing IT resources has a significant impact on the cloud properties displayed by different cloud deployment models, especially, regarding resource pooling, rapid elasticity and subsequently metered service (pay-per-use). A larger number of tenants reduces the effects of workload changes experienced by one tenant on the overall workload to be handled by the cloud provider. This is the case, because the overall workload subsumes all tenant workloads. By allowing fewer tenants to share a cloud or the underlying IT resources, the overall workload experienced by the cloud is less likely to be leveled out. The resulting workload changes make it harder for the cloud provider to ensure rapid elasticity, because resource pooling between tenants is less effective. This also affects the ability of the cloud provider to enable pay-per-use pricing models, because IT resources may become underutilized.

Figure 2.12 depicts the different cloud deployment models regarding the common level of elasticity and pay-per-use they provide. A *public cloud* (62) having the most tenants sharing it can enable the highest levels of elasticity and pay-per-use where only the operational costs are billed to customers. A *community cloud* (71) serves fewer tenants, often collaborating companies. An upfront investment may be required by these companies to establish the *community cloud* (71). Also, elasticity may be reduced as the collaborating companies may experience similar workloads. This effect is even more predominant in a *private cloud* (66) used by only one

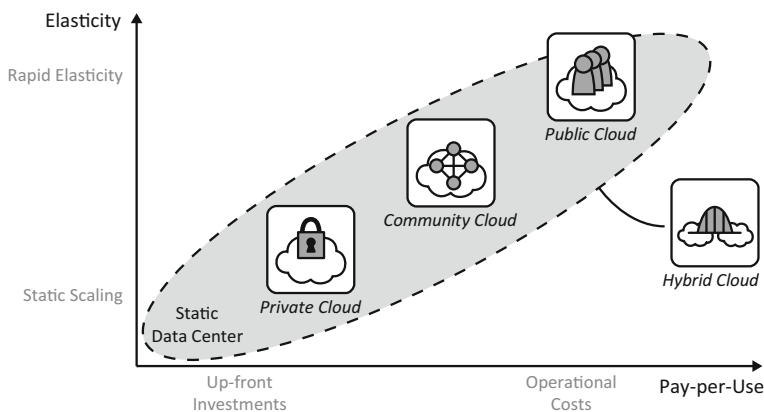


Fig. 2.12 Level of elasticity and pay-per-use of different cloud deployment types

tenant making upfront investments and reduced elasticity even more likely. A static data center is not covered by a pattern, but depicted in Fig. 2.12 as well. It does not use cloud computing technologies, requires up-front investments, and does not provide elasticity. The *hybrid cloud* (75) spans all these properties as it integrates applications hosted in the different environments. Note that the properties displayed by cloud deployment types are not generic. A *private cloud* accessed by a similar large and diverse user group as a *public cloud* is likely able to present the same properties. A *public cloud* used only by a small number of customers that experience similar workload will face similar challenges as a *private cloud*.

After reading this section you will be able to investigate which cloud deployment model is suitable for your application scenario. We cover each cloud deployment model in detail and how it deals with the essential cloud properties introduced in Sect. 1.1 on Page 3.

2.4.1 Public Cloud

IT resources are provided as a service to a very large customer group in order to enable elastic use of a static resource pool.



How can the cloud properties – on demand self-service, broad network access, pay-per-use, resource pooling, and rapid elasticity – be provided to a large customer group?

Context

A provider offering IT resources according to one of the cloud service models, *IaaS* (45), *PaaS* (49), or *SaaS* (55) has to maintain physical data centers that are limited in capacity. IT resources hosted in these static datacenters, nevertheless, shall be made accessible to tenants dynamically following a pay-per-use pricing model. The capacity of the data center, however, has to be planned statically and cannot be adjusted with the same elasticity, even though this behavior shall be displayed to customers. Enabling this elastic use of an otherwise static environment is especially challenging if customer experience changing workload described as *periodic workload* (29), *once-in-a-lifetime workload* (33), *unpredictable workload* (36), or *continuously changing workload* (40). Additional problems arise for the provider and the customer, as the different customers may not trust each other and, therefore, have to be isolated.

Solution

The *public cloud* is the cloud deployment model that best meets the desired cloud computing properties, because it serves a large number of customers and is, thus, large enough for customer diversity to level out peak workloads of individual applications. A *public cloud* shares a provided hosting environment between customers and this environment is accessible by many customers as depicted in Fig. 2.13 possibly reducing the costs for an individual customer. Leveraging economies of scale in this fashion, furthermore, enables a dynamic use of the static environment by customers, because workload peaks of some customers occur during times of low workload of other customers. The different workloads experienced by customers' applications are, therefore, leveled out regarding the overall customer group. This results in a more or less *static workload* (26) experienced by the provider that can be handled by a static number of physical IT resources in the

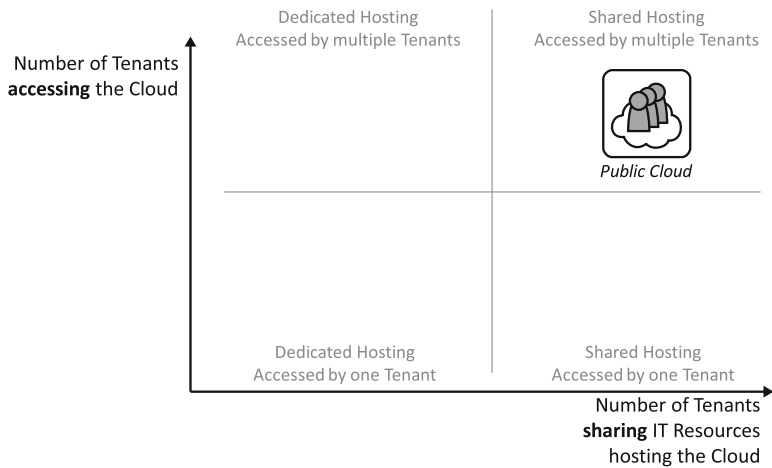


Fig. 2.13 Public cloud

data center. Security mechanisms are employed to isolate customers from each other. Often, this involves monitoring accesses and data entering and leaving the cloud in order to identify unlawful behavior. Such mechanisms are of vital importance to the success of the cloud provider, because trust is the major asset for the acceptance of *public clouds*.

Result

By sharing resources between a large number of customers and because of customer diversity, for example, due to geographic customer distribution, it is ensured that the peak workloads of customers can be handled, because other customers require fewer resources during those times. The size of a *public cloud* enables dynamic and elastic resource usage, while ensuring a leveled utilization of the static physical data center hosting the cloud. The capacity of this data center may be adjusted with much less dynamicity. The essential cloud computing properties introduced in Sect. 1.1 on Page 3 are enabled by *public clouds* as follows:

Access via network: *public clouds* are commonly made available over the Internet as they aim at being open to anyone without the need to install access software on the client. However, *public clouds* can also be available via a restricted and secured network connection.

On-demand self-service: in a *public cloud*, on-demand self-service is commonly realized as a Web-portal that allows customers to register with their credit card and then book, provision, manage, and decommission applications, platforms or infrastructure via that Web-portal. Also, an application programming interface

(API) is often provided to customers for automation of these tasks in their applications. Commonly, the allowed customer group is unrestricted enabling anybody with a credit card to sign up and use the *public cloud*. No complex setup or administration is needed on the provider side to accommodate a new customer. Customers can view their resource consumption and other monitoring information via the Web portal. The consequence of being able to rapidly sign up and then provision resources is that the Web portal and the cloud is seen as a standalone system that is not integrated in the order management and billing processes of the customer. As a consequence, even corporate users often use their private credit cards to book IT resources in *public clouds*.

Pay-per-use: *Public clouds* commonly do not require any upfront investments and bill strictly on a pay-per-use basis. Sometimes, cloud providers motivate long-term provisioning of IT resources by charging less for IT resources that are provisioned for a longer timeframe. This is beneficial for customers who may handle their *static workload* (26) with such resources and provision pay-per-use resources for workload peaks only. For the provider, the impact of workload changes on the customer side may be less and easier to predict by offering such pricing models. Some *public cloud* providers also offer cheaper resources during times when the overall utilization of the cloud is low in order to motivate time-uncritical applications to perform processing during these non-peak times and not during those times when the workload experienced by the cloud is high.

Resource pooling: *public clouds* are ideal with regard to resource pooling as they allow many customers to access and use the environment. These customers host different applications with versatile workload ensuring that workload peaks can be balanced across applications of multiple customers. As the cloud customer base and the users of applications hosted by these customers are large enough and provide enough diversity to level out peaks of individual applications, *public clouds* are often cheaper than *private clouds* (66) that sometimes have to deal with very similar usage by a few applications of a customer. One essential property that *public cloud* providers need to implement on their respective layer on the stack, i.e., the cloud service model they use, is multi-tenancy. As multiple customers (tenants) use the same IT resources, isolation is of crucial importance and must be guaranteed by the provider.

Rapid elasticity: *Public clouds* do not have any restrictions with regard to rapid elasticity as they are typically not integrated with customer-internal processes such as approval processes that sometimes hinder elasticity. Public clouds are often associated with unlimited resource availability, however, some cloud providers limit the amount of resources an individual consumer can host at the same time and implement higher quota only on request to accommodate customers with higher resource demands. The pricing models may be less dynamic than the degree of elasticity displayed by the cloud. Provisioned IT resources often become available within minutes and can be decommissioned with similar reactivity. However, the billing intervals may be longer, for example, resources may be billed for an hour even though they were provisioned only for 30 min.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): a *public cloud* offering *IaaS* provides a flexible and easy-to-use hosting environment for servers. Provider-supplied or customers-managed configurations of these servers, so called server images containing hardware configuration, operating systems, and pre-installed software, enable a quick provisioning of standardized servers. The physical hardware of *public IaaS clouds* is often virtualized through the use of *hypervisors* (101) to enable the sharing of physical hardware between customers. This resource pooling between customers eases elastic provisioning and decommissioning of servers and leads to low prices for customers all of which is relevant to provide the cloud properties covered in Sect. 1.1 on Page 3 to customers.
- *Platform as a Service (PaaS)* (49): a *public cloud* offering *PaaS* provides an open *elastic platform* (91) offering an *execution environment* (104) used by custom applications in addition to a number of offerings for communication and storage that hosted applications can use. These environments are often designed for publicly accessible Web-applications and provide functionality required in this scope.
- *Software as a Service (SaaS)* (55): *public clouds* offering *SaaS* provide standardized ready-to-use globally accessible applications over the Internet. These applications are commonly configurable in provider-defined means. The infrastructure and application components are highly shared with other customers to enable elasticity and pay-per-use. Typically, there are no upfront investments, but monthly fees per user accessing the application.

Known Uses

Public clouds providing *IaaS* are, for example, the Amazon Elastic Compute Cloud (EC2) [18], and Rackspace [19]. The Google App Engine [21], Microsoft Windows Azure [52], WSO2 Stratos Live [39], and Amazon Elastic Beanstalk [53] are examples for *public clouds* offering *PaaS* for applications developed in different programming languages. *Public clouds* using the *SaaS* service model are the Google Apps [49] or the Salesforce Customer Relationship Management (CRM) [45] offering. The Google Apps [49] provide collaboration tools, such as e-mail or scheduling. Salesforce also provides a *PaaS* (49) offering for extensions to its CRM *SaaS* offering, the Force [44] platform.

2.4.2 Private Cloud

IT resources are provided as a service exclusively to one customer in order to meet high requirements on privacy, security, and trust while enabling elastic use of a static resource pool as good as possible.



How can the cloud properties – on demand self-service, broad network access, pay-per-use, resource pooling, and rapid elasticity – be provided in environments with high privacy, security and trust requirements?

Context

Many factors, such as legal limitations, trust, and security regulations, motivate dedicated, company-internal hosting environments only accessible by employees and applications of a single company. These environments may follow any of the cloud service models: *IaaS* (45), *PaaS* (49), or *SaaS* (55). As they shall be used exclusively by one company, the challenge arises that physical data centers have a static capacity but shall provide a certain level of dynamicity and elasticity of IT resource use. Since the user group is, however, often smaller than in a *public cloud* (62) and often displays a similar workload behavior, economies of scale are harder to leverage.

Solution

A *private cloud* enables the cloud computing properties in a company-internal data center, thus, only one tenant accesses the cloud. Alternatively, the private cloud may be hosted exclusively in the data center of an external provider, then referred to as *outsourced private cloud* as depicted in Fig. 2.14 and according to the NIST cloud definition [3]. In case of an *outsourced private cloud*, some IT resources, such as networking infrastructure may be shared with other tenants. Sometimes, *public cloud* (62) providers offer means to create an isolated portion of their cloud made accessible to only one tenant, but still sharing hosting IT resources with all other tenants. This alternative, a *virtual private cloud*, is further described in the variation section.

Result

The main difference between a *private cloud* and other cloud deployment models is that the IT resources hosting the cloud that are shared with other customers are

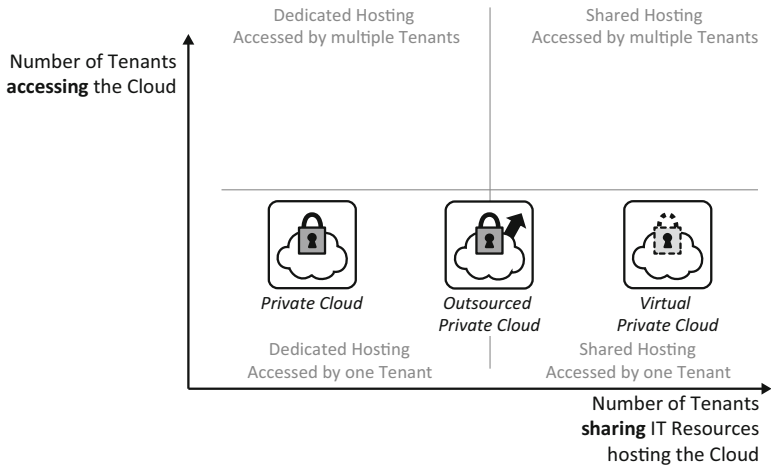


Fig. 2.14 Private cloud, outsourced private cloud, and virtual private cloud

reduced drastically, up to the point where no resources are shared. This separation may, especially, include the physical separation of networking hardware or the physical location of the data center as a whole. A *private cloud*, therefore, may provide a high level of security and privacy, which may, however, reduce its elasticity and the ability to provide a pay-per-use pricing model. This limitation is often caused by the smaller customer groups as, for example, in *public clouds* (62). The workloads of different tenants, thus, do not level each other out with respect to the workload experienced by the cloud as a whole. Static IT resources supporting the cloud then have to be provisioned for peak workloads as the peak workload of one tenant cannot be addressed with resources that another tenant does not need at that specific time. Of course, *private clouds* of a significantly large company whose employees display the same degree of diversity regarding workload behavior as customers of a *public cloud* (62) do not experience such limitations. Such very large *private clouds* can be considered equivalent to *public clouds* regarding the level of elasticity they provide. In this book, we mostly consider *private clouds* to be smaller than *public clouds* leading to the above mentioned challenges and limitations.

To fully support the cloud computing properties as a *public cloud* (62) does, a *private cloud* has to have a certain number of globally distributed users, so that the workload experienced by different departments is leveled out. But even if a *private cloud* does not exceed the critical size to leverage economies of scale effectively, the introduction of *private clouds* to a company can still be beneficial. A *private cloud* centralizes and standardizes IT resources to leverage economies of scale as good as possible and to enable automated management. This homogenization and automation helps companies to reduce their IT management costs, which are significantly inflicted by the complexity of heterogeneous IT environments [7].

The cloud computing properties introduced in Sect. 1.1 on Page 3 are enabled as follows.

Access via network: depending on the type of *private cloud* – normal or *outsourced private cloud*, access via the connection network is different. In a normal *private cloud* hosted in a company's own datacenter, access is enabled via the company-internal network. In an *outsourced private cloud* access is often realized via secured communication channel between the customer and provider networks, a so-called a virtual private network (VPN). A *virtual private cloud* uses the same connection network as the *public cloud* (62) offered by the cloud provider.

On-demand self-service: in a *private cloud* this property is commonly assured by a Web-portal providing a self-service interface similar to a *public cloud* (62). In difference to *public clouds*, customers do not sign up with their credit card but with their company-internal billing targets, such as project names or department identifiers. Furthermore, company-internal cost-management often requires that order management, approvals, and billing are integrated with the self-service interface to reflect the company-internal procedures. Especially, approval processes are often integrated into the self-service portal of a *private cloud*. In an *outsourced private cloud* setting in which integration with a customer's on-premise IT infrastructure is required, the provider's identity management, order management and billing systems are commonly integrated with the customer's systems. This integration can make the setup of such an *outsourced private cloud* quite complex.

Pay-per-use: in a *private cloud* where a set of IT resources is used only by one individual company and then elastically assigned to projects, applications or departments of that company, pay-per-use is often not easy to ensure. This is caused by the fact that all IT resource of the *private cloud* are provided exclusively for one company and, thus, experience the workload peaks of that company. Therefore, IT resources have to be provisioned for peak workloads if the company-internal diversity of user behavior is insufficient to level-out the utilization of the overall cloud. As a consequence, IT resources may experience times of low utilization hindering pay-per-use as these resources generate costs for the provider but are not used. In an *outsourced private cloud* the initial resource provisioning is delegated to the provider. The provider may assign some resources dynamically to this *outsourced private cloud*. However, it often has to be ensured at all times that no resources are shared between customers, which may increase the complexity of the process.

Resource pooling: key for effective resource pooling and, thus, the ability to scale elastically is the right mix of applications that allow balancing the peak workload while keeping overall resource utilization high. In a *private cloud*, resources cannot be shared between customers. The number of users sharing a resource pool is commonly reduced to the employees of the company using the *private cloud*. Therefore, resource pooling in a *private cloud* cannot be established on an intra-company basis, but has to occur on an intra-department or intra-application

basis by shifting IT resources between different departments, applications, or other organizational entities as the experienced workload demands. This may reduce the benefits of resource pooling to enable elasticity and pay-per-use in small *private clouds*.

Under these conditions, companies still benefit from the homogenization and centralization aspects of a *private cloud*. For example, a *private cloud* can provide standard development servers to reduce setup times for new projects.

Rapid elasticity: rapid elasticity works similarly in a *private cloud* as in a *public cloud* (62) regarding the enabling technologies. IT resources may, therefore, be provisioned and decommissioned very quickly via the self-service interface. There are two factors in a *private cloud* that may, however, hinder rapid elasticity: the limited size of data centers and a company's management processes involving human tasks. The former aspect means that elasticity in a *private cloud* may be limited if the whole company experiences a workload peak simultaneously, because rapid elasticity can only be performed within the boundaries of smaller static data centers. The second aspect considers the impact of a company's approval processes, billing processes etc. on provisioning and decommissioning times. For example, if a company requires a human manager to approve every provisioning of a new server for a project, this approval likely takes very long with respect to the actually required time to provision the server in the *private cloud*. Therefore, the integration of these processes with the self-service interface is of vital importance.

Variations

As a compromise between the flexibility of *public clouds* (62) and the security of *private clouds*, *public cloud* providers may offer so called *virtual private clouds*. These portions of a *public cloud* are often separated from other customers using the environment through networking and access configurations. This ensures that the *virtual private cloud* is isolated to some degree from other resources in the *public cloud* (62) while some IT resources may still be shared. Often, the *virtual private cloud* can additionally be integrated into the *private cloud* of a company using an encrypted communication link, such as a virtual private network (VPN) connection. The *virtual networking* (132) pattern discusses the used technologies offered by the *public cloud* provider as a communication offering in greater detail.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): a *private cloud* offering *IaaS* provides a flexible, isolated, and trusted hosting environment for servers accessible by one company. The provided virtual servers are standardized, as server images containing the server configuration, operating systems, and pre-installed software, may be shared between tenants, i.e., departments of the company. Multiple

servers may be provisioned based on a server image, as described by the *elastic infrastructure* (87) pattern. As hardware is not shared with others companies, the *private cloud* provides a high degree of isolation between the hosted servers but limited elasticity. Upfront investments in hardware may be necessary when establishing this deployment model.

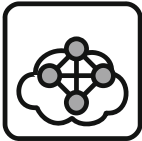
- *Platform as a Service (PaaS)* (49): the combination of *PaaS* and *private clouds* provides a standardized, isolated and trusted *execution environment* (104) for custom applications that provides common functionality for communication and storage used by these applications. While elasticity may still be limited, a high level of homogenization is realized by this environment as many applications of a company share it and rely on commonly accessible functions. Upfront investments may be required.
- *Software as a Service (SaaS)* (55): a *private cloud* following the *SaaS* service model offers trusted standardized applications to one company that are either standard applications or custom developed applications. This application is commonly used by many employees in a self-service manner and is integrated with other applications of the company. Similar to the other cloud service models using a *private cloud* may require upfront investments in infrastructure.
- *Batch processing component* (185): applications on a private *PaaS* or *IaaS* cloud may be designed to control the utilization of IT resources by delaying some of the workload processing. Especially, in a *private cloud*, where the overall number of IT resources that can be used elastically may be restricted, this allows processing workload when conditions are most feasible. *Batch processing components* (185) provide asynchronous application functionality that is accessible at all times, but is only processing requests when resources are available. Thus, the knowledge of the workload profiles of all applications hosted in a *private cloud* offering can lead to better resource utilization.

Known Uses

Tools to establish a private *IaaS* cloud in a company-internal datacenter are provided, for example, by the VMware product suite comprised of vCloud [20], vSphere [54], and ESX [54]. Open source software supporting similar functionality are Eucalyptus [37], OpenNebula [36], or OpenStack [35]. The mentioned variation of a *virtual private cloud* offering *IaaS* (45) is, for example, provided by Amazon Virtual Private Cloud (VPC) [55] or T-Systems' Dynamic Services for Infrastructure (DSI) [34].

2.4.3 Community Cloud

IT resources are provided as a service to a group of customers trusting each other in order to enable collaborative elastic use of a static resource pool.



How can the cloud properties – on demand self-service, broad network access, pay-per-use, resource pooling, and rapid elasticity – be provided to exclusively to a group of customers forming a community of trust?

Context

Companies may have to collaborate for various reasons. For example, a company may be a supplier of another company. Furthermore, a group of companies or public institutions, such as university or hospitals may have to exchange information or may have to share personnel for cost reduction. Whenever companies collaborate, they commonly have to access shared applications and data to do business. While these companies trust each other due to established contracts etc., the handled data and application functionality may be very sensitive and critical to their business, thus, rendering the use of an easily accessible *public cloud* (62) impossible. Also, *private clouds* (66) may be unsuitable, because IT resources are not used exclusively by one company but need to be accessed by the collaborating community.

Solution

IT resources required by all collaborating partners are offered in a controlled environment accessible only by the community of companies that generally trust each other. This *community cloud* contains all shared data and functionality that the participating companies need to do their business. Depending on the concrete requirements on privacy, security, and trust, a *community cloud* can be provided in a private data center controlled by the collaborating companies (often, one company acts as the cloud provider). Also, the *community cloud* be hosted in a data center of a third party, then referred to as *outsourced community cloud* shown in Fig. 2.15. Another variation is the hosting of a community cloud in an isolated portion of a *public cloud* (62) called *virtual community cloud*.

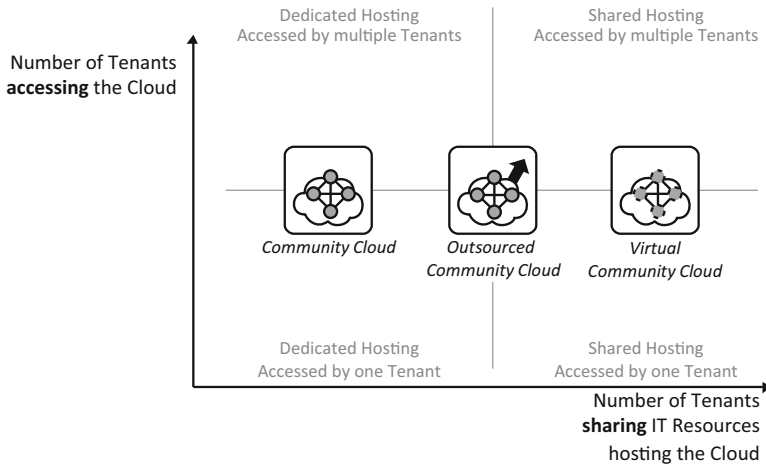


Fig. 2.15 Community cloud, outsourced community cloud, and virtual community cloud

Result

A *community cloud* hosted by one company is commonly used if this company has a central role in the collaboration, for example, a manufacturer provides a collaboration environment shared by its employees, contractors, and suppliers. *Outsourced community clouds* are, especially, suitable, if there is no company trusted enough by all the other companies to maintain the data center. *Virtual community clouds* are commonly easiest to establish as they depend on resources of a *public cloud* (62) to which access is only granted to collaborating partners. The cloud computing properties introduced in Sect. 1.1 on Page 3 are enabled by a *community cloud* as follows.

Access via network: similar to *public clouds* (62) and *private clouds* (66), the *community cloud* provides a Web-based self-service interface accessible by humans and commonly an applications programming interface (API) to be used for automation of provisioning and decommissioning of IT resources. If the *community cloud* is hosted by one company, that company may access it via its own network and grants external access to collaborating partners through a secured connection, i.e., via a virtual private network (VPN).

On-demand self-service: on-demand self-service within the *community cloud* follows the same principles as a *public cloud* (62). Any of the collaborating companies can provision and decommission IT resources, which are then billed to the ordering institution. The integration of company-internal billing and management processes is often not realized to the same degree as in a *private cloud* (66), because the individual processes of collaborating companies may differ significantly.

Pay-per-use: to establish a *community cloud*, up-front investments may be necessary for the collaborating companies. After its installment, the ability of the *community cloud* to provide pay-per-use to users mainly depends on its size and whether or not the workloads experienced by the collaborating companies lead to a leveled overall utilization of the cloud. This diversity may be less compared to customers of a *public cloud* (62), because collaborating companies may experience similar workload peaks due to tasks they perform together. The billing model supported by the *community cloud* needs to reflect that there are multiple companies that can order IT resources in the *community cloud*. This is similar to a *public cloud* (62), however, further billing models may be required. For example, one company may order IT resources to be used by its suppliers. Therefore, the tenant ordering the IT resources has no control over the users accessing them as would be the case if users were the employees of the ordering company. If IT resources are billed on a per-request basis, the ordering company may desire to define quotas on accesses performed by suppliers, which has to be supported by the *community cloud*.

Resource pooling: resources are shared between the collaborating companies. Therefore, the number tenants between which resources are shared is increased with respect to a *private cloud* (66), as sharing is possible on an inter-company basis. Because the collaborating companies trust each other more as users of a *public cloud* (62) due to established contracts etc. the desired isolation between customers of a community cloud may be reduced and, thus, easier to establish. Access control for the hosted IT resources, on the other hand, is likely to be more complex to enable the control of information exchange between companies.

Rapid elasticity: rapid elasticity in a community cloud is similar to that in a *public cloud* (62), if the customer group is large enough and displays a similar diversity. This helps to avoid the problems arising in a *private cloud* (66) due to the often smaller customer group. However, as the companies participating in a *community cloud* collaborate, they may experience similar workload behavior. The peaks of experienced *periodic workload* (29), *once-in-a-lifetime workload* (33), or *unpredictable workload* (36) may, therefore, occur at the same time hindering the ability of a *community cloud* to scale elastically.

Variations

Similar to the *virtual private cloud* variation of the *private cloud* (66) deployment model, a *community cloud* may be hosted in an isolated portion of a *public cloud* (62). This portion is commonly isolated on the networking level to avoid direct communication between the other IT resources in the *public cloud* (62) and those hosted in the *virtual community cloud*. Other hardware, for example, physical servers on which the virtual servers of multiple customers are hosted through the use of a *hypervisor* (101) may still be shared.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): *community PaaS clouds* provide a shared hosting environment where different companies may provision servers to provide data and shared functionality with each other. Preconfigured server images containing common operating systems and software combinations may be provided to provision servers in a more standardized manner, but the same level of homogenization as in *private clouds* (66) is unlikely as companies can have their own internal standardization efforts to homogenize IT resources that they also use in the *community cloud*.
- *Platform as a Service (PaaS)* (49): a *community cloud* offering *PaaS* provides a standardized, collaborative trusted *elastic platform* (91) providing an *execution environment* (104) and other offerings based on which collaborating companies may develop shared applications that rely on common functionality and data. The elasticity of this platform may be limited if the number of collaborating companies is low, but it is highly standardized to ease development and to coordinate information exchange. Often, these environments focus on the collaborative collection and evaluation of data.
- *Software as a Service (SaaS)* (55): *community SaaS clouds* often provide the same functionality as public *SaaS* offerings, but application functionality is only accessible to collaborating companies. Another difference to other *SaaS* offerings is that collaborating companies do not desire to perceive the application as if they were the only user, but want to exchange information with other companies using the *SaaS* offering, for example, to handle orders or to manage schedules.

Known Uses

The same technologies to create *private IaaS clouds* can be used to create *community IaaS clouds* in a dedicated data center. The only difference is the group of companies that are granted access to this environment. Google provides its Google Apps [49] public *SaaS* offering also as a *SaaS community cloud* for U.S. government agencies [56].

2.4.4 Hybrid Cloud

Different clouds and static data centers are integrated to form a homogeneous hosting environment.



How can the cloud properties – on demand self-service, broad network access, pay-per-use, resource pooling, and rapid elasticity – be provided across clouds and other environments?

Context

The cloud deployment model used by a company in a specific use case is often determined by the required level of accessibility, privacy, security and trust, because *private clouds* (66), *public clouds* (62), and *community clouds* (71) significantly differ on these assurances. A company is, however, likely to use a large set of applications to support its business. These applications and possibly their individual application components may have versatile requirements making different cloud deployment models suitable to host them. In order to match these requirements efficiently, it is, therefore, desirable to host applications and their components in different clouds and static data centers leading to an integration challenge. Even if applications could use the same cloud, legacy and non-cloud applications may exist that also have to interact with cloud applications. A company, therefore, has to manage these different hosting environments and communication has to be enabled between them.

Solution

A *hybrid cloud* integrates multiple *private clouds* (66), *public clouds* (62), *community clouds* (71) and static data centers. Applications and their individual components are deployed to the hosting environment best suited for their requirements and interconnection of these environments is ensured. A *hybrid cloud*, therefore, integrates different hosting environments that can be accessed by different number of tenants and share underlying IT resources between different amounts of tenants as shown in Fig. 2.16.

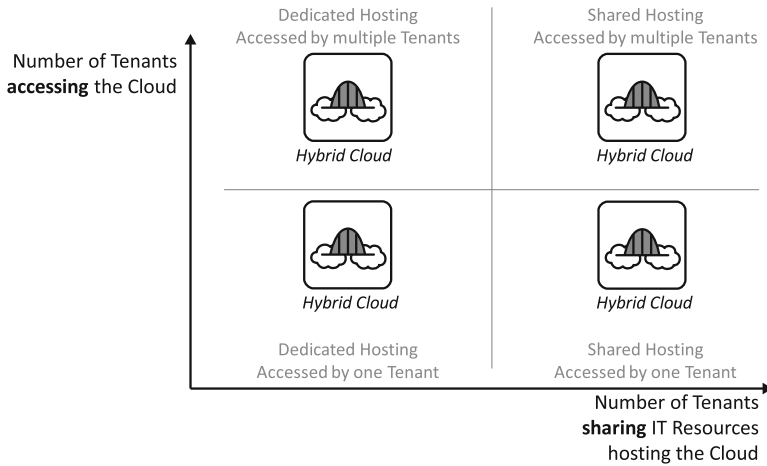


Fig. 2.16 Hybrid cloud in the cloud scope

Result

Through the establishment of a *hybrid cloud*, the applications and their components required by a company may be hosted in multiple hosting environments, for example, to use a *private cloud* (66) to handle *static workload* (26) and deploy additional IT resources in a *public cloud* (62), if the workload increases suddenly, for example, due to a *once-in-a-lifetime workload* (33) peak. Such a workload spill-over allows *private clouds* (66) to be scaled more optimistically resulting in a higher utilization and a reduced cost of the overall environment. The cloud computing properties introduced in Sect. 1.1 on Page 3 are ensured by integrating the individual environments to form the *hybrid cloud* as follows. Especially, a *hybrid cloud* can be used to add some of these properties to a static data center by integrating it with a cloud environment.

Access via network: access via network to the separate clouds and static data centers is enabled as if they were not integrated. Additional connections are established between these environments to enable access between them as well. As the environments may assure different levels of privacy, security, and trust, this access has to be regulated in many cases. For example, *public clouds* (62) and *community clouds* (71) could be integrated with a *private cloud* (66). The *community cloud* (71) and *public cloud* (62) are accessible via a more or less public network that often can be accessed easily from the *private cloud* (66). Access from the public network to the *private cloud* is, however, often restricted. Under such conditions, a *private cloud* (66) commonly acts as the resource broker of the formed *hybrid cloud* that decides for each access request to a resource whether it is served from *private cloud* (66) resources, or *public cloud* (62) resources. This is determined by the required service level agreements, especially taking security and trust issues into account.

On-demand self-service: similar to an integrated access via network, the different self-service interfaces of integrated clouds can be subsumed to provide a unified interface to customers. Often, one cloud or data center acts as the provider of the self-service Web-portal and forwards provisioning and decommissioning requests to the other clouds if needed. The known uses section of this pattern covers some tools providing such unified interfaces to multiple clouds.

Pay-per-use: as the self-service interface of a *hybrid cloud* integrates different clouds, it has to consolidate the resource billing of the different environments as well. If possible, a *hybrid cloud* should provide a homogeneous billing model rather than just adapting the billing models of all integrated hosting environments to reduce complexity. Customers are then billed according to this billing model and the *hybrid cloud* provider pays integrated providers according their individual billing models.

Resource pooling: sharing of resources is enabled internally by the integrated *private clouds* (66), *public clouds* (62), or *community clouds* (71). However, the *hybrid cloud* may add additional management functionality on top of this resource pooling by optimizing the IT resource distribution among the integrated environments. For example, the *hybrid cloud* may consider the individual utilization of integrated environments in this decision, thus, providing a homogenous resource pool comprised of individual resource pools of different hosting environments. This integration of provisioning and decommissioning decisions has to be considered especially, when integrating a static environment that does not offer resource pooling with a cloud environment.

Rapid elasticity: elasticity regarding the overall resources in all environments should be enabled by the *hybrid cloud's* integration functionality just as resource pooling has to be enabled between the integrated hosting environments. Again, the integrated environments handle their individual elasticity and may support more or less flexibility regarding the provisioning and decommissioning of IT resources. Through this integration, a *hybrid cloud* may, for example, provision IT resources in a *public cloud* (62), if those of a static data center are insufficient during workload peaks. For this decision, the *hybrid cloud*, thus, has to monitor the integrated environments to balance resources among them.

Related Patterns

- *Infrastructure as a Service (IaaS)* (45): integration of different clouds on the *IaaS* level is commonly done by enabling networking communication between different clouds. One approach is to establish virtual private networks (VPN) between the different clouds which may make the deployment of additional servers in these environments handling the integration necessary. The involved technologies are covered in greater detail by the *virtual networking* (132) pattern.

- *Platform as a Service (PaaS)* (49): integration of different *PaaS* offerings may be enabled by special application components hosted in these environments. The *application component proxy* (228) pattern describes how application functionality may be made available in a different environment than where it is hosted. The *message mover* (225) pattern describes how messaging functionality provided in different environments may be integrated.
- *Software as a Service (SaaS)* (55): many *SaaS* applications allow customers to integrate other applications that they use in different environments. This functionality is, however, supported by the provider of the *SaaS* application. If custom developed integration functionality is required, for example, to trigger a company-internal application if a certain condition arises in a *SaaS* application custom implementations are likely to be required. Many *SaaS* providers offer *PaaS* (49) offerings for this purpose, where customers may host custom extensions to the *SaaS* application, i.e., to integrate them with other applications used by the customer.
- *Hybrid cloud applications* (starting in Sect. 6.3 on Page 303): these patterns cover different distributions of application functionality providing user interfaces, processing functionality, and data handling among different environments. The motivation to assign certain functionality to an environment is discussed as well as the integration of this functionality into one application.

Known Uses

Challenges to integrate communication between IT resources residing in different clouds have to be addressed to form a *hybrid cloud*. This can be done using an Enterprise Service Bus (ESB) that offers accesses to different resources in a seamless fashion. On-premise ESBs, such as Apache ServiceMix [57] or IBM WebSphere Enterprise Service Bus [58] can be used. Additionally, the ESB itself can be accessed as a service from a cloud providing a *PaaS* (49) offering. For example, this is offered by Microsoft AppFabric, part of Windows Azure [52]. VMWare and its partners, such as T-Systems [59], offer the VMware vCloud product that can be used to build *private clouds* (66) in a model, where the *private cloud* can be extended into a hosted cloud offering of the partner to build a *hybrid cloud*. In addition to the communication between the IT resources, the management interfaces have to be integrated by a *hybrid cloud*. Apache Deltacloud [60] offers the integration of different clouds behind one application programming interface. A similar approach is taken by Apache Libcloud [61] and Jclouds [62] offering the integration of multiple clouds in a programming library.

Cloud Computing Patterns

Fundamentals to Design, Build, and Manage Cloud
Applications

Fehling, C.; Leymann, F.; Retter, R.; Schupeck, W.;
Arbitter, P.

2014, XXVI, 367 p., Hardcover

ISBN: 978-3-7091-1567-1