2.2 Understanding the different classifications of clouds

Now that you've learned about the technological underpinnings of cloud computing, such as virtualization, elasticity, storage, and databases, it's useful to understand how those concepts are employed in the different types (classifications) of cloud computing services being offered. Let's go back to the taxonomy of cloud types from chapter 1—IaaS, PaaS, and DaaS—to classify the clouds from the most prominent providers in the industry.

2.2.1 Amazon EC2: Infrastructure as a Service

Amazon EC2 is categorized as IaaS (some cloud observers call it HaaS, but Amazon has added so many additional services that Hardware as a Service would now be a misnomer). It was the first and is by far the biggest in this category. Amazon opened its service in 2006 after initially using excess capacity from its retail operation. The company claimed to have over 500,000 users by the end of 2008.

Amazon EC2 is the most general purpose of the major clouds but has the least support for automatic scaling or failover, both of which have to be programmed into the application. This is in contrast to the automatic and invisible scaling that occurs in the PaaS types of clouds, such as Google's AppEngine, which we'll discuss in section 2.2.3. In IaaS-type clouds, such as EC2, elasticity requires careful programming using their APIs. On the other hand, you can use any programming language, and you have complete control over your application in an IaaS cloud. Sure, it requires more manual work, but you get something that has the appearance of being physical hardware that you have control over from the operating system outward. The LAMP stack is the easiest and most common EC2 configuration (see table 2.6).

Table 2.6 The components of the LAMP stack in an IaaS cloud

L	Linux	Operating system
A	Apache	Web server
M	MySQL	Relational database
P	PHP	Server side of website

Amazon EC2 and Xen paravirtualization

Amazon EC2 utilizes a customized version of the open source Xen hypervisor, taking advantage of paravirtualization. Because paravirtualized guest OSs rely on the hypervisor to provide support for operations that normally require privileged access, the guest OS runs with no elevated access to the CPU.

Paravirtualization is more efficient than a virtualized environment where the guest OS runs unmodified. But the OS must be ported to the paravirtualized environment so that certain OS tasks that would have to be performed by the VMM and run more slowly can be directly executed by the guest OS. This is why Amazon doesn't run any OS you may desire—it runs only OSs that it or the original vendor has ported and fully tested.

Amazon has an extensive API for all its services, some of which are described in table 2.7. It has a SOAP as well as a simple HTML (`GET`, `POST`) form for its APIs. The company needs only a dozen and a half calls to request and configure virtualized hardware in the cloud.

Table 2.7 Other Amazon cloud services (effectively providing some PaaS capabilities)

Service	Description
Simple Storage Service (S3)	Cloud storage used to store and retrieve large amounts of data from anywhere on the web through a simple API. Well integrated with EC2: AMIs are stored in S3, and data transferred from S3 to EC2 doesn't invoke separate charges.

Table 2.7 Other Amazon cloud services (effectively providing some PaaS capabilities) (*continued*)

Service	Description
SimpleDB	Provides the core database functions of indexing (special organizational entities for faster lookups) and querying. Avoids the big expense of relational database licensing, the requisite DBA, and the complex setup. But it isn't a relational database, has no schema, and doesn't work with SQL.
CloudFront	A web service for content delivery that competes with Akamai. Provides an easy way to distribute content to end users with low latency and high data-transfer speeds in a pay-as-you-go model.
Simple Queue Service (SQS)	A hosted queue for storing messages as they travel between computers. Useful for moving data between distributed components of applications that perform different tasks, without losing messages or requiring each component to be always available.

EC2 pricing starts at roughly a nickel per small Linux-based instance (CPU) hour, up to about half a dollar on a high-end Linux instance.[6] S3 pricing is about $0.15 per GB per month, scaling downward as more storage is used.

2.2.2 *Microsoft Azure: Infrastructure as a Service*

Microsoft's Azure is IaaS in the same way as Amazon EC2, but it also has other services that operate more at the PaaS level. Many of Microsoft's end-user applications are being recast to run in the cloud. As a result, increasingly, this overall platform is trying to reach the SaaS level to head off Google's thrust against Microsoft Office with the Google Docs and Google Apps SaaS offerings.

The box labeled Windows Azure in figure 2.6 is Windows Server 2008 modified to run in the cloud environment. This means it was paravirtualized to make it run efficiently in the virtualized environment created by running Microsoft's Hypervisor on bare hardware in Microsoft's cloud data centers.

Internally, the OS layer—derived from Windows Server 2008—consists of four pillars: storage (like a file system); the fabric controller, which is a management system for modeling/deploying and provisioning; virtualized computation/VM; and a development environment, which allows developers to emulate Windows Azure on their desktop and plug in Visual Studio, Eclipse, or other tools to write cloud applications against it. Because of this architecture, you merely have to deploy Azure on a single machine; then, you can duplicate multiple instances of it on the rest of the servers in the cloud using virtualization technology.

[6] http://aws.amazon.com/ec2/pricing/

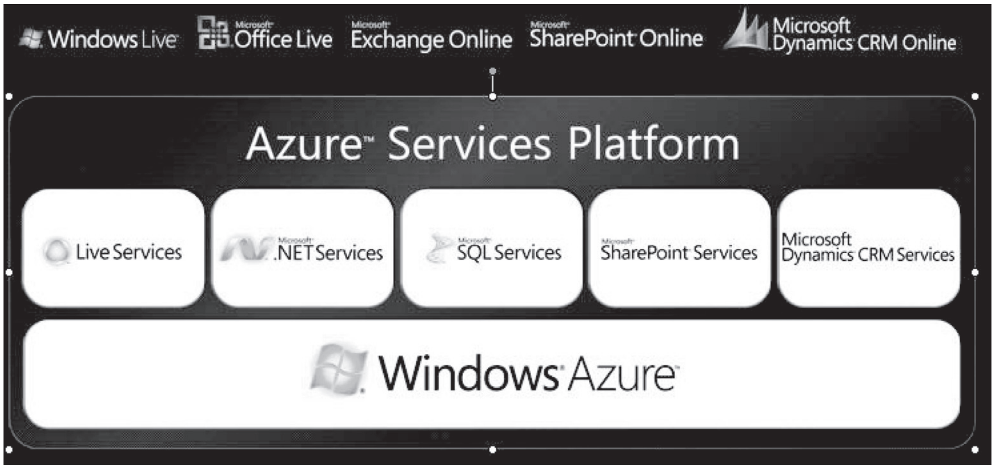
Windows Live
Microsoft Office Live
Microsoft Exchange Online
Microsoft SharePoint Online
Microsoft Dynamics CRM Online
Azure Services Platform
Live Services
Microsoft .NET Services
Microsoft SQL Services
Microsoft SharePoint Services
Microsoft Dynamics CRM Services
Windows Azure

Figure 2.6 The Windows Azure architecture and framework. At the bottom level is the Windows Azure operating system. This runs in the virtualization environment created by Microsoft's Hypervisor running on bare hardware. At the top layer are end-user applications, which Microsoft is recasting to be delivered as SaaS. Source: Microsoft.

Applications for Microsoft's Azure are written in proprietary programming environments like Visual Studio using the .NET libraries, and compiled to the Common Language Runtime, Microsoft's language-independent managed environment.

WINDOWS AZURE API

The Windows Azure API is a REST-based API that uses X.509 client certificates for authentication. Table 2.8 lists a portion of the API, giving you an idea of how applications running in the Azure cloud are manipulated and controlled. This is the set of calls related to performing operations on hosted services.

Similar to Amazon, a set of building-block services run on top of Azure creating Microsoft's PaaS capabilities. The initial set of services includes the following:

- Live Services
- SQL Services
- .Net Services
- SharePoint Services
- CRM Services

You can treat these lower-level services as APIs—they have no user interface elements—when constructing cloud applications.

Azure pricing is comparable to Amazon with computing time set at $0.12 per hour, storage at $0.15 per GB, and storage transactions at $0.01 per 10 KB. For the structured database, fees for the web edition are set at up to 1 GB relational database at $9.99 per month and for the business edition up to 10 GB relational database at $99.99 per month. A tiered, all-you-can-eat (within limits) model is said to be coming.

Table 2.8 A portion of the RESTful Windows Azure API

Service	Description
List Hosted Services	Lists the hosted services available under the current subscription.

`GET`
`https://management.core.windows.net/<subscription-id>/services/`
`hostedservices`	

Get Hosted Service Properties	Retrieves system properties for the specified hosted service. These properties include the service name and service type; the name of the affinity group to which the service belongs, or its location if it isn't part of an affinity group; and, optionally, information about the service's deployments.

`GET`
`https://management.core.windows.net/<subscription-id>/services/`
`hostedservices/`
`<service-name>`	

Create Deployment	Uploads a new service package and creates a new deployment on staging or production.

`POST`
`https://management.core.windows.net/<subscription-id>/services/`
`hostedservices/`
`<service-name>/deploymentslots/<deployment-slot-name>`	

Get Deployment	May be specified as follows. Note that you can delete a deployment either by specifying the deployment slot (staging or production) or by specifying the deployment's unique name.

`GET`
`https://management.core.windows.net/<subscription-id>/services/`
`hostedservices/`
`<service-name>/deploymentslots/<deployment-slot>/`
`GET`
`https://management.core.windows.net/<subscription-id>/services/`
`hostedservices/`
`<service-name>/deployments/<deployment-name>/`	

Swap Deployment	Initiates a virtual IP swap between the staging and production deployment slots for a service. If the service is currently running in the staging environment, it's swapped to the production environment. If it's running in the production environment, it's swapped to staging. This is an asynchronous operation whose status must be checked using Get Operation Status.

`POST`
`https://management.core.windows.net/<subscription-id>/hostedservices/`
`<service-name>`	

Table 2.8 A portion of the RESTful Windows Azure API (*continued*)

Service	Description
Delete Deployment	Deletes the specified deployment. This is an asynchronous operation.

```
DELETE
https://management.core.windows.net/<subscription-id>/services/
   hostedservices/
     <service-name>/deploymentslots/<deployment-slot>
DELETE
https://management.core.windows.net/<subscription-id>/services/
   hostedservices/
     <service-name>/deployments/<deployment-name>
```

2.2.3 Google App Engine: Platform as a Service

App Engine is a pure PaaS cloud targeted exclusively at traditional web applications, enforcing an application structure of clean separation between a stateless computation tier and a stateful storage tier. The virtualization and the elasticity that are so visible in the IaaS model are almost completely invisible here. But they're a big part of the picture behind the scenes. One of the selling propositions of this model is its *automatic* elasticity in the face of capacity requirement changes.

The App Engine programming languages are Python and Java. App Engine isn't suitable for general-purpose computing. It works best for web applications and relies on the assumption of a request-reply structure, which assumes long periods of no CPU utilization (such as, human think time). Consequently, Google can and does severely ration CPU time for each request.

App Engine's automatic scaling and high-availability mechanisms, and the proprietary MegaStore data storage (built on BigTable) available to App Engine applications, all rely on these constraints. But if your application fits within those constraints, there is probably no faster and cheaper way to build an application that scales automatically and runs on the largest cloud on the planet.

APP ENGINE DEVELOPMENT ENVIRONMENT

The App Engine development environment consists of these elements:

- *Sandbox*—Applications run in a secure environment that provides limited access to the underlying operating system. These limitations allow App Engine to distribute web requests for the application across multiple servers and to start and stop servers to meet traffic demands. The sandbox isolates your application in its own secure, reliable environment independent of the hardware, operating system, and physical location of the web server.

- *Java runtime environment*—You can develop applications for the Java 6 runtime environment using common Java web development tools and API standards. An app interacts with the environment using the Java Servlet standard and can use common web application technologies, such as JavaServer Pages (JSPs). Apps access most App Engine services using Java standard APIs. The environment includes the Java SE Runtime Environment (JRE) 6 platform and libraries. The restrictions of the sandbox environment are implemented in the JVM. An app can use any JVM bytecode or library feature, as long as it doesn't exceed the sandbox restrictions.
- *Python runtime environment*—You can develop applications using the Python 2.5 programming language and run it on a Python interpreter. App Engine includes APIs and tools for Python web application development, including a data-modeling API, a web application framework, and tools for managing and accessing the App's data. The Python environment includes the Python standard library within the limitations of the sandbox environment. Application code written for the Python environment must be written exclusively in Python. The Python environment provides APIs for the datastore, Google Accounts, URL fetch, and email services.
- *Datastore*—App Engine provides a distributed data-storage service that features a query engine and transactions. This distributed datastore scales with the application's needs automatically. As we discussed previously regarding cloud databases, the App Engine datastore isn't like a traditional relational database. Data objects, or *entities*, have a kind and a set of properties. Queries can retrieve entities of a given kind filtered and sorted by the values of the properties. Property values can be of any of the supported property value types. Datastore entities are *schemaless*. The application code enforces and provides the structure of data entities. The Java JDO/JPA interfaces and the Python datastore interface include features for applying and enforcing this structure.

App Engine is free under these daily thresholds: 6.5 hours of CPU time, and 1 GB of data transferred in and out of the application. Beyond this, outgoing bandwidth costs $0.12 per GB, incoming bandwidth costs $0.10 per GB, CPU time is $0.10 per hour, stored data is $0.15 per GB per month, and recipients emailed are $0.0001 per recipient.

2.2.4 Ruby on Rails in a cloud: Platform as a Service

Ruby on Rails (RoR) is an open-source web application framework for the Ruby programming language. It's intended to be used with an agile development methodology, often used by web developers due to its suitability for short, client-driven projects. Similar to Google's App Engine, RoR applications are limited to request-response architecture web applications.

OPEN SOURCE SOFTWARE Computer software available in source code form for which the source code and certain other rights normally reserved for copyright holders are provided under a software license that permits users to study, change, and improve the software. Some consider open source a philosophy; others consider it a pragmatic methodology. Before the term *open source* became widely adopted, developers and producers used a variety of phrases to describe the concept; *open source* gained hold with the rise of the internet and the attendant need for massive retooling of the computing source code. Open-source software is most often developed in a public, collaborative manner.

The Ruby language was designed to combine Smalltalk's conceptual elegance, Python's ease of use and learning, and Perl's pragmatism. Many teams experience 10X faster development of web applications using RoR. But many have reported significant challenges getting RoR to scale massively, which probably has to do with architecture and design choices made in the application as opposed to something endemic to RoR itself.

Many small companies jumped in early to offer RoR stacks that run on top of Amazon's EC2, including Heroku, Aptana, EngineYard, and others.

2.2.5 Salesforce.com's Force.com: Platform as a Service

Salesforce.com is the most successful SaaS application used in the enterprise. It's a customer-relationship-management (CRM) application that has run strictly as a cloud application since 1999.

Force.com is the company's PaaS capability, where developers use the Apex programming language to create add-on applications that integrate into the main Salesforce application and are hosted on Salesforce.com's cloud infrastructure.

Google and Salesforce have created an integration between App Engine and Force.com such that applications can be built using either environment and still access the stored repository of corporate data on Salesforce's site.

Force.com also runs an exchange called AppExchange; it's a directory of applications built for Salesforce by third-party developers, which users can purchase and add to their Salesforce environment. More than 800 applications are available from over 450 ISVs.

The Force.com list price is $5.00 per login with a maximum of five logins per user per month. According to the company's website, "Force.com cloud pricing is for occasional-use, widely-deployed apps and is available for platform use only and not for CRM applications."

Our last classification is a strange one—strange because it's not about a different type of application environment but instead is about a different ownership structure. So-called Datacenter as a Service is about private companies creating a *private* cloud just for their use.

2.2.6 Private clouds: Datacenter as a Service (DaaS)

Private cloud (also *internal cloud* and *corporate cloud*) is a term for a computing architecture that provides hosted services to a limited number of people behind a firewall. The same advances in virtualization, automation, and distributed computing that enable the cloud for Amazon, Microsoft, and Google have allowed corporate network and data-center administrators to effectively become service providers that meet the needs of their customers within the corporation.

The concept of a private cloud is designed to appeal to an organization that needs or wants more control over its data than it can get by using a third-party hosted service, such as Amazon's EC2 or S3. Internal IT providers that build private clouds have to make fundamental changes in their operations so they behave and provide benefits (on a smaller scale) similar to those of cloud computing providers. In addition to economic gains through higher utilization and a pay-for-what-you-use model, an enterprise, to enable the private cloud model, implements changes in operations which, at the very least, makes an organization better equipped to shift to or overflow to a public cloud when appropriate.

The contrarian view

Here's the rub: some say private clouds are expensive data centers with a fancy name. Pundits predict that within the next year or so, we'll have seen the rise and fall of this concept. Whereas everyone agrees that virtualization, service-oriented architectures, and open standards are all great things for companies operating a data center to consider, critics argue that all this talk about private clouds is a distraction from the real news: the vast majority of companies shouldn't need to worry about operating any sort of data center anymore, cloud-like or not.

SOME CONCERNS FOR THOSE THINKING ABOUT PRIVATE CLOUDS

If you're considering implementing a private cloud, keep the following in mind:

- *Private clouds are small scale.* There's a reason why most innovative cloud computing providers have their roots in powering consumer web technology—that's where the numbers are. Few corporate data centers will see anything close to the type of volume seen by these vendors. And volume drives cost savings through the huge economies of scale we've discussed.
- *Legacy applications don't cloudify easily.* Legacy applications moved to a private cloud will see marginal improvements at best. You can achieve only so much without re-architecting these applications to a cloud infrastructure.
- *On-premises doesn't mean more secure.* The biggest drivers toward private clouds have been fear, uncertainty, and doubt about security. For many, it feels more secure to have your data behind your firewall in a data center that you control. But unless your company spends more money and energy thinking about security than Amazon, Google, and Salesforce, that is a fallacy.
- *Do what you do best.* There's no simple set of tricks that an operator of a data center can borrow from Amazon or Google. These companies make their living operating the world's largest data centers. They're constantly optimizing how

> they operate based on real-time performance feedback from millions of transactions. You can try to learn from and emulate them, but your rate of innovation will never be the same—private clouds will always be many steps behind the public clouds.

AMAZON VIRTUAL PRIVATE CLOUD

Amazon Virtual Private Cloud (Amazon VPC) is a secure and seamless bridge between a company's existing IT infrastructure and the AWS cloud. Although it isn't a private cloud as we defined it, this approach offers corporations a hybrid model merging aspects of their data center with Amazon's cloud.

Amazon VPC enables an enterprise to connect its existing infrastructure to a set of isolated AWS compute resources via a Virtual Private Network (VPN) connection and to extend existing management capabilities, such as security services, firewalls, and intrusion-detection systems, to include AWS resources. You'll learn much more about cloud security, private clouds, and VPC in chapter 4.

Until now, we've explored the technological underpinnings of clouds to understand how they work, and we've applied that knowledge to a few of the most prominent clouds in a variety of categories to understand how they compare and contrast. You're now informed enough to ask this question: What type of cloud do I need?