Module - 5

Cloud Platforms in Industry. (Chapter - 9)

9. Cloud Platforms in Industry

An overview of a few prominent cloud computing platforms and a brief description of the types of service they offer.

A cloud computing system can be developed using either a single technology and vendor or a combination of them.

This chapter presents some of representative cloud computing solutions offered as Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) services in the market.

9.1 Amazon web services

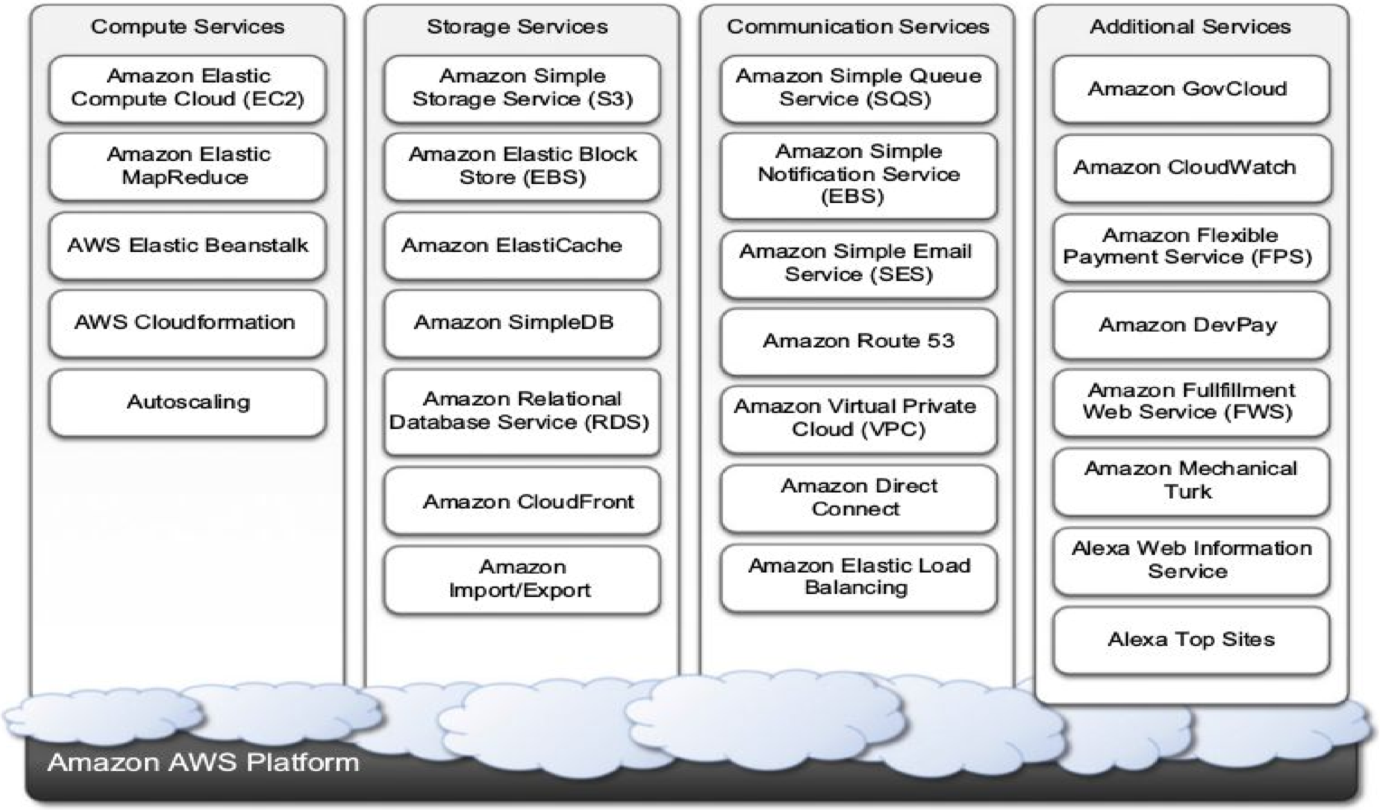
9.1.1 Compute services
9.1.2 Storage services
9.1.3 Communication services

9.1 Amazon web services

Amazon Web Services (AWS) is a platform that allows the development of flexible applications by providing solutions for elastic infrastructure scalability, messaging, and data storage.

The platform is accessible through SOAP or RESTful Web service interfaces and provides a Web-based console where users can handle administration and monitoring of the resources required, as well as their expenses computed on a pay-as-you-go basis.

Figure 9.1 shows all the services available in the AWS ecosystem. At the base of the solution stack are services that provide raw compute and raw storage: Amazon Elastic Compute (EC2) and Amazon Simple Storage Service (S3).



Compute Services
Amazon Elastic Compute Cloud (EC2)
Amazon Elastic MapReduce
AWS Elastic Beanstalk
AWS Cloudformation
Autoscaling
Storage Services
Amazon Simple Storage Service (S3)
Amazon Elastic Block Store (EBS)
Amazon ElastiCache
Amazon SimpleDB
Amazon Relational Database Service (RDS)
Amazon CloudFront
Amazon Import/Export
Communication Services
Amazon Simple Queue Service (SQS)
Amazon Simple Notification Service (EBS)
Amazon Simple Email Service (SES)
Amazon Route 53
Amazon Virtual Private Cloud (VPC)
Amazon Direct Connect
Amazon Elastic Load Balancing
Additional Services
Amazon GovCloud
Amazon CloudWatch
Amazon Flexible Payment Service (FPS)
Amazon DevPay
Amazon Fullfillment Web Service (FWS)
Amazon Mechanical Turk
Alexa Web Information Service
Alexa Top Sites
Amazon AWS Platform

FIGURE 9.1

Amazon Web Services ecosystem.

9.1.1 Compute services

The fundamental service in this space is Amazon EC2, which delivers an IaaS solution that has served as a reference model for several offerings from other vendors in the same market segment.

Amazon EC2 allows deploying servers in the form of virtual machines created as instances of a specific image.

Images come with a preinstalled operating system and a software stack, and instances can be configured for memory, number of processors, and storage.

Users are provided with credentials to remotely access the instance and further configure or install software if needed.

1. Amazon machine images
2. EC2 instances
3. EC2 environment
4. Advanced compute services

1. Amazon machine images

Amazon Machine Images (AMIs) are templates from which it is possible to create a virtual machine.

They are stored in Amazon S3 and identified by a unique identifier in the form of ami-xxxxxx and a manifest XML file.

AMI contains physical file system layout with a predefined operating system installed as Amazon Ramdisk Image (ARI, id: ari-yyyyyy) and the Amazon Kernel Image (AKI, id: aki-zzzzzz)

A common practice is to prepare new AMIs to create an instance from a preexisting AMI, log into it once it is booted and running, and install all the software needed. Using the tools provided by Amazon, we can convert the instance into a new image. Once an AMI is created, it is stored in an S3 bucket and the user can decide whether to make it available to other users or keep it for personal use.

2. EC2 instances

EC2 instances represent virtual machines. They are created using AMI as templates, which are specialized by selecting the number of cores, their computing power, and the installed memory. The processing power is expressed in terms of virtual cores and EC2 Compute Units (ECUs).

the use of ECUs helps give users a consistent view of the performance offered by EC2 instances.

One ECU is defined as giving the same performance as a 1.0 - 1.2 GHz 2007 Opteron or 2007 Xeon processor.

We can identify six major configurations for EC2 instances.

- **Standard instances.** offers set of configurations that are suitable for most applications. EC2 provides three different categories of increasing computing power, storage, and memory.
- **Micro instances.** suitable for those applications that consume limited amount of computing power and memory and need bursts in CPU cycles to process surges in workload.
- **High-memory instances.** targets applications that need to process huge workloads and require large amounts of memory. Three-tier Web applications characterized by high traffic are the target profile.
- **High-CPU instances.** targets compute-intensive applications.
- **Cluster Compute instances.** used to provide virtual cluster services. Instances in this category are characterized by high CPU compute power and large memory and an extremely high I/O and network performance, which makes it suitable for HPC applications.
- **Cluster GPU instances.** provides instances featuring graphic processing units (GPUs) and high compute power, large memory, and extremely high I/O and network performance. This class is particularly suited for cluster applications that perform heavy graphic computations.

3. EC2 environment

EC2 instances are executed within a virtual environment. The EC2 environment is in charge of allocating addresses, attaching storage volumes, and configuring security in terms of access control and network connectivity.

It is possible to associate an Elastic IP to each instance. Elastic IPs allow instances running in EC2 to act as servers reachable from the Internet.

EC2 instances are given domain name in the form ec2-xxx-xxx-xxx.compute-x.amazonaws.com,

where xxx-xxx-xxx normally represents the four parts of the external IP address separated by a dash, and compute-x gives information about the availability zone where instances are deployed.

Currently, there are five availability zones: two in the United States (Virginia and Northern California), one in Europe (Ireland), and two in Asia Pacific (Singapore and Tokyo).

4. Advanced compute services

AWS CloudFormation constitutes an extension of the simple deployment model that characterizes EC2 instances. CloudFormation introduces the concepts of templates, which are JSON formatted text files that describe the resources needed to run an application or a service in EC2 together.

Templates provide a simple and declarative way to build complex systems and integrate EC2 instances with other AWS services such as S3, SimpleDB, SQS, SNS, Route 53, Elastic Beanstalk, and others.

AWS Elastic Beanstalk constitutes a simple and easy way to package applications and deploy them on the AWS Cloud. This service simplifies the process of provisioning instances and deploying application code and provides appropriate access to them.

Currently, this service is available for Web applications developed with the Java/Tomcat technology stack.

Beanstalk simplifies tedious tasks without removing the user's capability of accessing—and taking over control of—the underlying EC2 instances.

Amazon Elastic MapReduce provides AWS users with a cloud computing platform for MapReduce applications. It utilizes Hadoop as the MapReduce engine, deployed on a virtual infrastructure composed of EC2 instances, and uses Amazon S3 for storage needs.

Elastic MapReduce introduces elasticity and allows users to dynamically size the Hadoop cluster according to their needs, as well as select the appropriate configuration of EC2 instances to compose the cluster.

9.1.2 Storage services

The core service is represented by Amazon Simple Storage Service (S3). This is a distributed object store that allows users to store information in different formats. The core components of S3 are two: buckets and objects. Buckets represent virtual containers in which to store objects; objects represent the content that is actually stored. Objects can also be enriched with metadata that can be used to tag the stored content with additional information.

1 S3 key concepts
2 Amazon elastic block store
3 Amazon ElastiCache
4 Structured storage solutions
5 Amazon CloudFront

1 S3 key concepts

S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface.

- The storage is organized in a two-level hierarchy.
- Stored objects cannot be manipulated like standard files.
- Content is not immediately available to users.
- Requests will occasionally fail.

Access to S3 is provided with RESTful Web services. These express all the operations that can be performed on the storage in the form of HTTP requests (GET, PUT, DELETE, HEAD, and POST).

Resource naming

Buckets, objects, and attached metadata are made accessible through a REST interface. Therefore, they are represented by uniform resource identifiers (URIs) under the s3.amazonaws.com domain.

Amazon offers three different ways of addressing a bucket:

1. Canonical form: http://s3.amazonaws.com/bukect_name/
2. Subdomain form: http://bucketname.s3.amazon.com/
3. Virtual hosting form: http://bucket-name.com/

Buckets

A bucket is a container of objects. It can be thought of as a virtual drive hosted on the S3 distributed storage, which provides users with a flat store to which they can add objects. Buckets are top-level elements of the S3 storage architecture and do not support nesting. That is, it is not possible to create "subbuckets" or other kinds of physical divisions.

Objects and metadata

Objects constitute the content elements stored in S3. Users either store files or push to the S3 text stream representing the object's content. An object is identified by a name that needs to be unique within the bucket in which the content is stored. The name cannot be longer than 1,024 bytes when encoded in UTF-8, and it allows almost any character. Buckets do not support nesting.

Access control and security

Amazon S3 allows controlling the access to buckets and objects by means of Access Control Policies (ACPs). An ACP is a set of grant permissions that are attached to a resource expressed by means of an XML configuration file.

A policy allows defining up to 100 access rules, each of them granting one of the available permissions to a grantee.

Currently, five different permissions can be used:

A. READ allows the grantee to retrieve an object and its metadata and to list the content of a bucket as well as getting its metadata.
B. WRITE allows the grantee to add an object to a bucket as well as modify and remove it.
C. READ_ACP allows the grantee to read the ACP of a resource.
D. WRITE_ACP allows the grantee to modify the ACP of a resource.
E. FULL_CONTROL grants all of the preceding permissions.

2 Amazon elastic block store

The Amazon Elastic Block Store (EBS) allows AWS users to provide EC2 instances with persistent storage in the form of volumes that can be mounted at instance startup. They accommodate up to 1 TB of space and are accessed through a block device interface, thus allowing users to format them according to the needs of the instance they are connected to.

EBS volumes normally reside within the same availability zone of the EC2 instances that will use them to maximize the I/O performance. It is also possible to connect volumes located in different availability zones. Once mounted as volumes, their content is lazily loaded in the background and according to the request made by the operating system. This reduces the number of I/O requests that go to the network.

3 Amazon ElastiCache

ElastiCache is an implementation of an elastic in-memory cache based on a cluster of EC2 instances.

It provides fast data access through a Memcached-compatible protocol so that applications can transparently migrate to ElastiCache.

ElastiCache is based on a cluster of EC2 instances running the caching software, which is made available through Web services.

An ElastiCache cluster can be dynamically resized according to the demand of the client applications.

4 Structured storage solutions

Amazon provides applications with structured storage services in three different forms:

- Preconfigured EC2 AMIs,
- Amazon Relational Data Storage (RDS), and
- Amazon SimpleDB.

Preconfigured EC2 AMIs are predefined templates featuring an installation of a given database management system. EC2 instances created from these AMIs can be completed with an EBS volume for storage persistence. Available AMIs include installations of IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Sybase, and Vertica.

RDS is relational database service that relies on the EC2 infrastructure and is managed by Amazon. Developers do not have to worry about configuring the storage for high availability, designing failover strategies, or keeping the servers up-to-date with patches. Moreover, the service provides users with automatic backups, snapshots, point-in-time recoveries, and facilities for implementing replications.

Amazon SimpleDB is a lightweight, highly scalable, and flexible data storage solution for applications that do not require a fully relational model for their data. SimpleDB provides support for semistructured data, the model for which is based on the concept of domains, items, and attributes.

SimpleDB uses domains as top-level elements to organize a data store. These domains are roughly comparable to tables in the relational model. Unlike tables, they allow items not to have all the same column structure; each item is therefore represented as a collection of attributes expressed in the form of a key-value pair.

5 Amazon CloudFront

CloudFront is an implementation of a content delivery network on top of the Amazon distributed storage infrastructure. It leverages a collection of edge servers strategically located around the globe to better serve requests for static and streaming Web content so that the transfer time is reduced.

AWS provides users with simple Web service APIs to manage CloudFront. To make available content through CloudFront, it is necessary to create a distribution. This identifies an origin server, which contains the original version of the content being distributed, and it is referenced by a DNS domain under the Cloudfront.net domain name.

The content that can be delivered through CloudFront is static (HTTP and HTTPS) or streaming (Real Time Messaging Protocol, or RMTP).

9.1.3 Communication services

Amazon provides facilities to structure and facilitate the communication among existing applications and services residing within the AWS infrastructure. These facilities can be organized into two major categories:
1. Virtual networking and
2. Messaging.

1. Virtual networking
Virtual networking comprises a collection of services that allow AWS users to control the connectivity to and between compute and storage services.
Amazon Virtual Private Cloud (VPC) and Amazon Direct Connect provide connectivity solutions in terms of infrastructure.
Amazon VPC provides flexibility in creating virtual private networks within the Amazon infrastructure and beyond. The service providers prepare templates for network service for advanced configurations. Templates include public subnets, isolated networks, private networks accessing Internet through network address translation (NAT), and hybrid networks including AWS resources and private resources.
Amazon Direct Connect allows AWS users to create dedicated networks between the user private network and Amazon Direct Connect locations, called ports. This connection can be further partitioned in multiple logical connections and give access to the public resources hosted on the Amazon infrastructure. The advantage is the consistent performance of the connection between the users premises and the Direct Connect locations.
Amazon Route 53 implements dynamic DNS services that allow AWS resources to be reached through domain names different from the amazon.com domain. By leveraging the large and globally distributed network of Amazon DNS servers.
2. Messaging.
The three different types of messaging services offered are

- Amazon Simple Queue Service (SQS),
- Amazon Simple Notification Service (SNS), and
- Amazon Simple Email Service (SES).

Amazon SQS constitutes disconnected model for exchanging messages between applications by means of message queues, hosted within the AWS infrastructure. Using the AWS console or directly the underlying Web service AWS, users can create an unlimited number of message queues and configure them to control their access. Applications can send messages to any queue they have access to. These messages are securely and redundantly stored within the AWS infrastructure for a limited period of time, and they can be accessed by other (authorized) applications.
Amazon SNS provides a publish-subscribe method for connecting heterogeneous applications. Amazon SNS allows applications to be notified when new content of interest is available. This feature is accessible through a Web service whereby AWS users can create a topic, which other applications can subscribe.
Amazon SES provides AWS users with a scalable email service that leverages the AWS infrastructure. Once users are signed up for the service, they have to provide an email that SES will use to send emails on their behalf. To activate the service, SES will send an email to verify the given address and provide the users with the necessary information for the activation.

9.2 Google AppEngine

Google AppEngine is a PaaS implementation.
Distributed and scalable runtime environment that leverages Google's distributed infrastructure to scale out applications.

9.2.1 Architecture and core concepts

AppEngine is a platform for developing scalable applications accessible through the Web. **Figure 9.2.**
The platform is logically divided into four major components: infrastructure, the runtime environment, the underlying storage, and the set of scalable services.

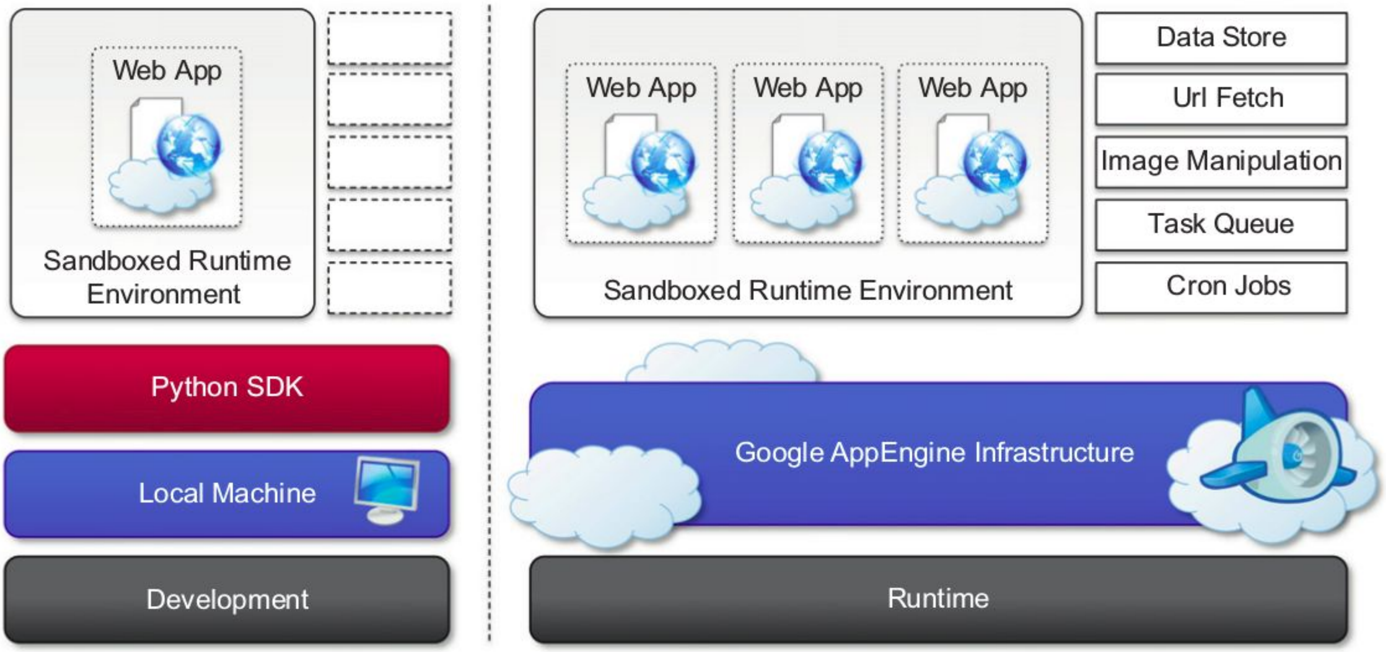


FIGURE 9.2

Google AppEngine platform architecture.

1 Infrastructure

AppEngine hosts Web applications, and its primary function is to serve users requests efficiently.

AppEngine's infrastructure takes advantage of many servers available within Google datacenters. For each HTTP request, AppEngine locates the servers hosting the application that processes the request, evaluates their load, and, if necessary, allocates additional resources or redirects the request to an existing server.

The infrastructure is also responsible for monitoring application performance and collecting statistics on which the billing is calculated.

2 Runtime environment

The runtime environment represents the execution context of applications hosted on AppEngine.

Sandboxing- One of the major responsibilities of the runtime environment is to provide the application environment with an isolated and protected context in which it can execute without causing a threat to the server and without being influenced by other applications. In other words, it provides applications with a sandbox.

If an application tries to perform any operation that is considered potentially harmful, an exception is thrown and the execution is interrupted.

Supported runtimes- Currently, it is possible to develop AppEngine applications using three different languages and related technologies: Java, Python, and Go.

AppEngine currently supports Java 6, and developers can use the common tools for Web application development in Java, such as the Java Server Pages (JSP), and the applications interact with the environment by using the Java Servlet standard.

Support for Python is provided by an optimized Python 2.5.2 interpreter. As with Java, the runtime environment supports the Python standard library.

Developers can use a specific Python Web application framework, called webapp, simplifying the development of Web applications.

The Go runtime environment allows applications developed with the Go programming language to be hosted and executed in AppEngine. Currently the release of Go that is supported by AppEngine is r58.1. The SDK includes the compiler and the standard libraries for developing applications in Go and interfacing it with AppEngine services.

3 Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data.

Static file servers- Web applications are composed of dynamic and static data. Dynamic data are a result of the logic of the application and the interaction with the user. Static data often are mostly constituted of the components that define the graphical layout of the application or data files.

DataStore- DataStore is a service that allows developers to store semistructured data. The service is designed to scale and optimized to quickly access data. DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key.

DataStore imposes less constraint on the regularity of the data but, at the same time, does not implement some of the features of the relational model.

The underlying infrastructure of DataStore is based on Bigtable, a redundant, distributed, and semistructured data store that organizes data in the form of tables.

DataStore provides high-level abstractions that simplify interaction with Bigtable. Developers define their data in terms of entity and properties, and these are persisted and maintained by the service into tables in Bigtable.

DataStore also provides facilities for creating indexes on data and to update data within the context of a transaction. Indexes are used to support and speed up queries. A query can return zero or more objects of the same kind or simply the corresponding keys.

4 Application services

Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in Web applications

UrlFetch - The sandbox environment does not allow applications to open arbitrary connections through sockets, but it does provide developers with the capability of retrieving a remote resource through HTTP/HTTPS by means of the UrlFetch service. Applications can make synchronous and asynchronous Web requests and integrate the resources obtained in this way into the normal request- handling cycle of the application.

UrlFetch is not only used to integrate meshes into a Web page but also to leverage remote Web services in accordance with the SOA reference model for distributed applications.

MemCache- This is a distributed in-memory cache that is optimized for fast access and provides developers with a volatile store for the objects that are frequently accessed. The caching algorithm implemented by MemCache will automatically remove the objects that are rarely accessed. The use of MemCache can significantly reduce the access time to data; developers can structure their applications so that each object is first looked up into MemCache and if there is a miss, it will be retrieved from DataStore and put into the cache for future lookups.

Mail and instant messaging- AppEngine provides developers with the ability to send and receive mails through Mail. The service allows sending email on behalf of the application to specific user accounts. It is also possible to include several types of attachments and to target multiple recipients.

AppEngine provides also another way to communicate with the external world: the Extensible Messaging and Presence Protocol (XMPP). Any chat service that supports XMPP, such as Google Talk, can send and receive chat messages to and from the Web application, which is identified by its own address.

Account management- AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts.

Using Google Accounts, Web applications can conveniently store profile settings in the form of key-value pairs, attach them to a given Google account, and quickly retrieve them once the user authenticates.

5 Compute services

AppEngine offers additional services such as Task Queues and Cron Jobs that simplify the execution of computations.

Task queues- A task is defined by a Web request to a given URL, and the queue invokes the request handler by passing the payload as part of the Web request to the handler. It is the responsibility of the request handler to perform the "task execution," which is seen from the queue as a simple Web request.

Cron jobs- the required operation needs to be performed at a specific time of the day, which does not coincide with the time of the Web request. In this case, it is possible to schedule the required operation at the desired time by using the Cron Jobs service.

9.2.2 Application life cycle

AppEngine provides support for all the phases characterizing the life cycle of an application: testing and development, deployment, and monitoring.

1 Application development and testing

Developers can start building their Web applications on a local development server.

The development server simulates the AppEngine runtime environment by providing a mock implementation of DataStore, MemCache, UrlFetch, and the other services leveraged by Web applications.

AppEngine builds indexes for each of the queries performed by a given application in order to speed up access to the relevant data. This capability is enabled by a priori knowledge about all the possible queries made by the application; such knowledge is made available to AppEngine by the developer while uploading the application.

Java SDK- The Java SDK provides developers with the facility for building applications with the Java 5 and Java 6 runtime environments. Using the Eclipse software installer, it is possible to download and install Java SDK, Google Web Toolkit, and Google AppEngine plug-ins into Eclipse. These three components allow developers to program powerful and rich Java applications for AppEngine.

Python SDK- The Python SDK allows developing Web applications for AppEngine with Python 2.5. It provides a standalone tool, called GoogleAppEngineLauncher, for managing Web applications locally and deploying them to AppEngine.

The Python implementation of the SDK also comes with an integrated Web application framework called webapp that includes a set of models, components, and tools that simplify the development of Web applications and enforce a set of coherent practices.

The webapp framework has been reimplemented and made available in the Python SDK so that it can be used with AppEngine.

2 Application deployment and management

Once application has been developed and tested, it can be deployed on AppEngine with simple click or command-line tool. Before performing such task, it is necessary to create application identifier, which will be used to locate application from Web browser by typing the address http://<application-id>.appspot.com.

An application identifier is mandatory because it allows unique identification of the application while it's interacting with AppEngine. Developers use an app identifier to upload and update applications.

Once an application identifier has been created, it is possible to deploy an application on AppEngine. This task can be done using either respective development environment (GoogleAppEngineLauncher and Google AppEngine plug-in) or the command-line tools.

9.2.3 Cost model

Once application has been tested and tuned for AppEngine, it is possible to set up a billing account and obtain more allowance and be charged on a pay-per-use basis. This allows developers to identify appropriate daily budget that they want to allocate for given application.

An application is measured against **billable quotas, fixed quotas, and per-minute quotas.**

Google AppEngine uses these quotas to ensure that users do not spend more than the allocated budget and that applications run without being influenced by each other from a performance point of view.

Billable quotas identify the daily quotas that are set by application administrator and are defined by daily budget allocated for application.

Free quotas are part of the billable quota and identify the portion of the quota for which users are not charged.

Fixed quotas are internal quotas set by AppEngine that identify infrastructure boundaries and define operations that application can carry out on infrastructure.

9.2.4 Observations

AppEngine, a framework for developing scalable Web applications, leverages Google's infrastructure.

The core components of service are scalable and sandboxed runtime environment for executing applications and a collection of services that implement most of the common features.

One of the characteristic elements of AppEngine is use of simple interfaces that allow applications to perform specific operations that are optimized and designed to scale.

Building on top of these blocks, developers can build applications and let AppEngine scale them out when needed.

9.3 Microsoft Azure

Microsoft Windows Azure is a cloud operating system built on top of Microsoft datacenters' infrastructure and provides developers with a collection of services for building applications with cloud technology.

Services range from compute, storage, and networking to application connectivity, access control, and business intelligence.

Figure 9.3 provides an overview of services provided by Azure. These services can be managed and controlled through the Windows Azure Management Portal, which acts as an administrative console for all the services offered by the Azure platform.

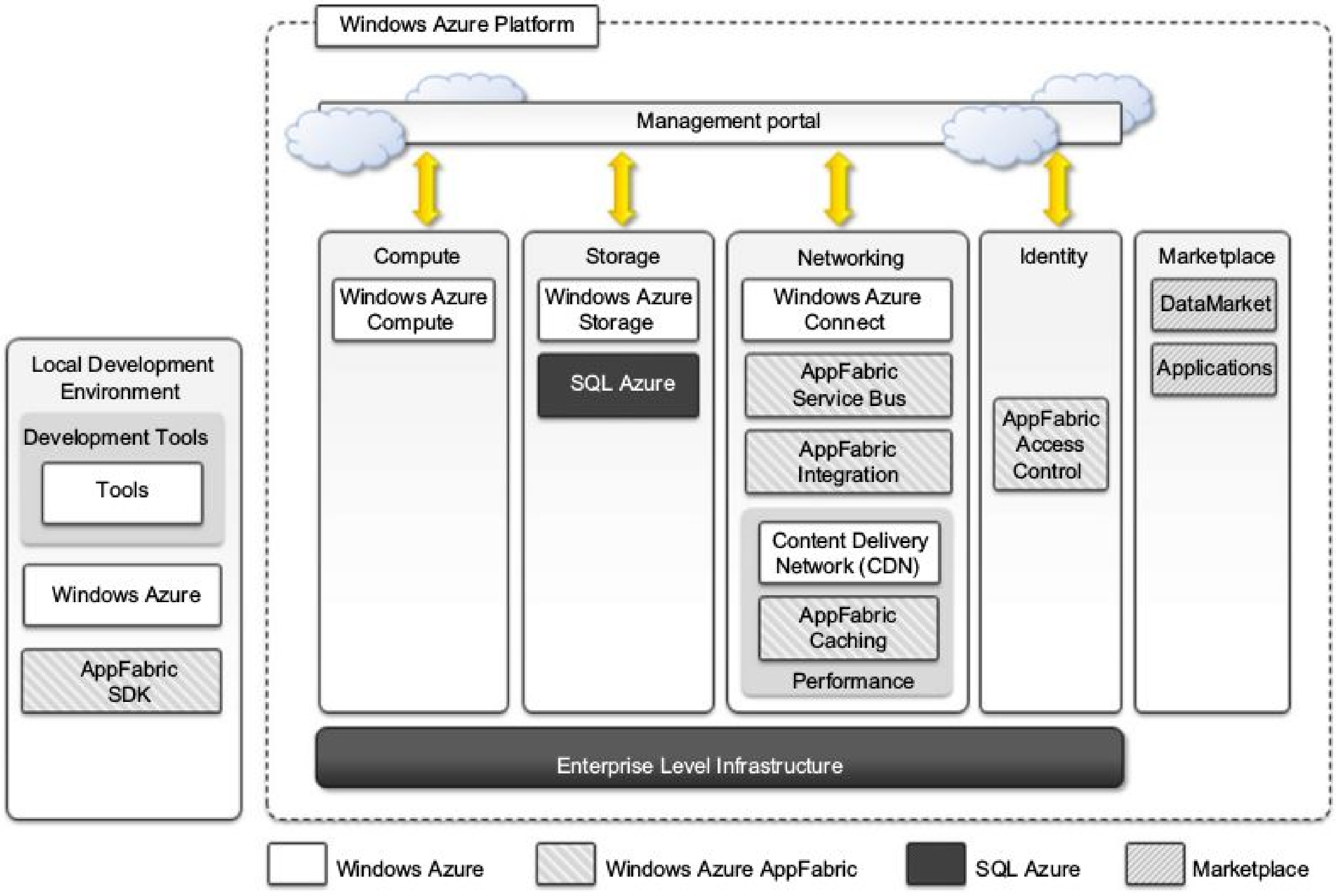


FIGURE 9.3

Microsoft Windows Azure Platform Architecture.

9.3.1 Azure core concepts

The Windows Azure platform is made up of a foundation layer and a set of developer services that can be used to build scalable applications. These services cover compute, storage, networking, and identity management, which are tied together by middleware called AppFabric.

1 Compute services

Compute services are the core components of Microsoft Windows Azure, and they are delivered by means of the abstraction of roles.

Currently, there are three different roles: Web role, Worker role, and Virtual Machine (VM) role.

Web role- The Web role is designed to implement scalable Web applications. Web roles represent the units of deployment of Web applications within the Azure infrastructure. They are hosted on the IIS 7 Web Server.

Since version 3.5, the .NET technology natively supports Web roles; developers can directly develop their applications in Visual Studio, test them locally, and upload to Azure.

Web roles can be used to run and scale PHP Web applications on Azure (CGI Web Role).

Worker role - Worker roles are designed to host general compute services on Azure. They can be used to quickly provide compute power or to host services that do not communicate with the external world through HTTP. A common practice for Worker roles is to use them to provide background processing for Web applications developed with Web roles.

A Worker role runs continuously from the creation of its instance until it is shut down. The Azure SDK provides developers with convenient APIs and libraries that allow connecting the role with the service provided by the runtime and easily controlling its startup as well as being notified of changes in the hosting environment.

Virtual machine role- The Virtual Machine role allows developers to control computing stack of their compute service by defining a custom image of the Windows Server 2008 R2 operating system and all the service stack required by their applications. The Virtual Machine role is based on the Windows Hyper-V virtualization technology.

Developers can image a Windows server installation complete with all the required applications and components, save it into a Virtual Hard Disk (VHD).

2 Storage services

Compute resources are equipped with local storage in the form of a directory on the local file system.

Windows Azure provides different types of storage solutions that complement compute services with a more durable and redundant option.

Blobs

Azure allows storing large amount of data in the form of binary large objects (BLOBs) by means of the blobs service.

Block blobs- Block blobs are composed of blocks and are optimized for sequential access; therefore they are appropriate for media streaming. Currently, blocks are of 4 MB, and a single block blob can reach 200 GB in dimension.

Page blobs- Page blobs are made of pages that are identified by offset from beginning of blob. A page blob can be split into multiple pages or constituted of single page. This type of blob is optimized for random access and can be used to host data different from streaming. Maximum dimension of page blob can be 1TB.

Azure drive

Page blobs can be used to store an entire file system in the form of a single Virtual Hard Drive (VHD) file. This can then be mounted as a part of the NTFS file system by Azure compute resources, thus providing persistent and durable storage.

Tables

Tables constitute a semistructured storage solution, allowing users to store information in the form of entities with a collection of properties. Entities are stored as rows in the table and are identified by a key, which also constitutes the unique index built for the table. Users can insert, update, delete, and select a subset of the rows stored in the table.

Currently, a table can contain up to 100 TB of data, and rows can have up to 255 properties, with a maximum of 1 MB for each row. The maximum dimension of a row key and partition keys is 1 KB.

Queues

Queue storage allows applications to communicate by exchanging messages through durable queues, thus avoiding lost or unprocessed messages. Applications enter messages into a queue, and other applications can read them in a first-in, first-out (FIFO) style.

3 Core infrastructure: AppFabric

AppFabric is a comprehensive middleware for developing, deploying, and managing applications on the cloud or for integrating existing applications with cloud services.

AppFabric implements an optimized infrastructure supporting scaling out and high availability; sandboxing and multitenancy; state management; and dynamic address resolution and routing.

Access control- AppFabric provides the capability of encoding access control to resources in Web applications and services into a set of rules that are expressed outside the application code base. These rules give a great degree of flexibility in terms of the ability to secure components of the application and define access control policies for users and groups.

Service bus - Service Bus constitutes the messaging and connectivity infrastructure provided with AppFabric for building distributed and disconnected applications The service is designed to allow transparent network traversal and to simplify the development of loosely coupled applications, without renouncing security and reliability and letting developers focus on the logic of the interaction rather than the details of its implementation. Service Bus allows services to be available by simple URLs, which are untied from their deployment location.

Azure cache- Windows Azure provides a set of durable storage solutions that allow applications to persist their data. Azure Cache is a service that allows developers to quickly access data persisted on Windows Azure storage or in SQL Azure. The service implements a distributed in-memory cache of which the size can be dynamically adjusted by applications according to their needs.

4 Other services

Windows Azure virtual network- Networking services for applications are offered under the name Windows Azure Virtual Network, which includes Windows Azure Connect and Windows Azure Traffic Manager. Windows Azure Connect allows easy setup of IP-based network connectivity among machines hosted on the private premises and the roles deployed on the Azure Cloud. This service is particularly useful in the case of

VM roles.

Windows Azure content delivery network- Windows Azure Content Delivery Network (CDN) is the content delivery network solution that improves the content delivery capabilities of Windows Azure Storage and several other Microsoft services, such as Microsoft Windows Update and Bing maps.

9.3.2 SQL Azure

SQL Azure is a relational database service hosted on Windows Azure and built on the SQL Server technologies. The service extends the capabilities of SQL Server to the cloud and provides developers with a scalable, highly available, and fault-tolerant relational database. SQL Azure is accessible from either the Windows Azure Cloud or any other location that has access to the Azure Cloud. It is fully compatible with the interface exposed by SQL Server, so applications built for SQL Server can transparently migrate to SQL Azure. **Figure 9.4** shows the architecture of SQL Azure. Access to SQL Azure is based on the Tabular Data Stream (TDS) protocol, which is the communication protocol underlying all the different interfaces used by applications to connect to a SQL Server-based installation such as ODBC and ADO.NET.

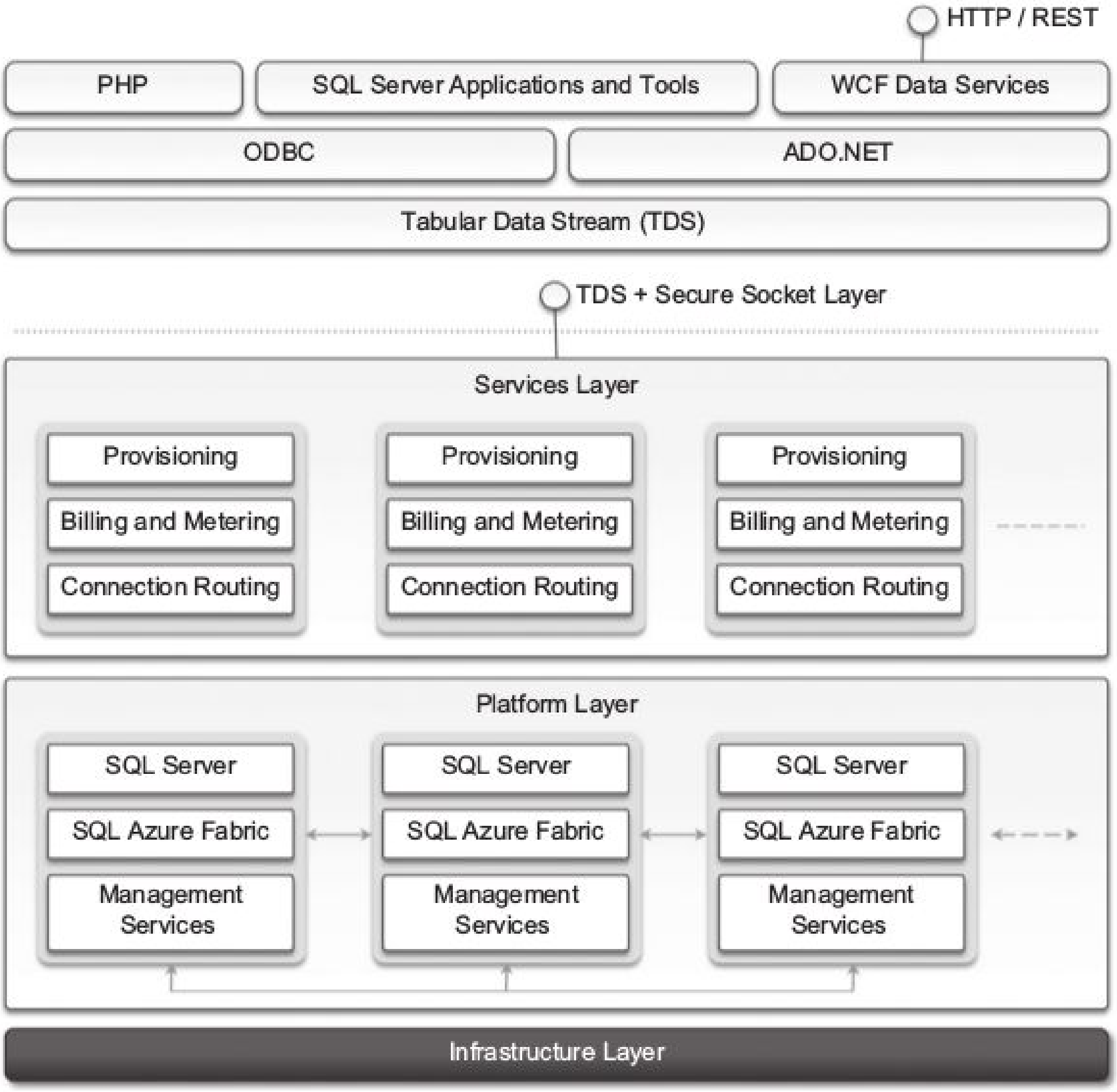


FIGURE 9.4

SQL Azure architecture.

Developers have to sign up for a Windows Azure account in order to use SQL Azure. Once the account is activated, they can either use the Windows Azure Management Portal or the REST APIs to create servers and logins and to configure access to servers.

SQL Azure servers are abstractions that closely resemble physical SQL Servers: They have a fully qualified domain name under the database.windows.net (i.e., server-name.database.windows.net) domain name. This simplifies the management tasks and the interaction with SQL Azure from client applications.

Currently, two different editions are available: Web Edition and Business Edition. The former is suited for small Web applications and supports databases with a maximum size of 1 GB or 5 GB. The latter is suited for independent software vendors, line-of-business applications, and enterprise applications and supports databases with a maximum size from 10 GB to 50 GB, in increments of 10 GB.

9.3.3 Windows Azure platform appliance

The Windows Azure platform can also be deployed as an appliance on third-party data centers and constitutes the cloud infrastructure governing the physical servers of the datacenter. The Windows Azure Platform Appliance includes Windows Azure, SQL Azure, and Microsoft- specified configuration of network, storage, and server hardware. The appliance is a solution that targets governments and service providers who want to have their own cloud computing infrastructure.

9.3.4 Observations

Windows Azure is Microsoft's solution for developing cloud computing applications. Azure is an implementation of the PaaS layer and provides the developer with a collection of services and scalable middleware hosted on Microsoft datacenters that address compute, storage, networking, and identity management needs of applications.

The core components of the platform are composed of compute services, storage services, and middleware. Compute services are based on the abstraction of roles, which identify a sandboxed environment where developers can build their distributed and scalable components.

SQL Azure is another important element of Windows Azure and provides support for relational data in the cloud. SQL Azure is an extension of the capabilities of SQL Server adapted for the cloud environment and designed for dynamic scaling.