业务定乾坤

— Amazon 云计算服务剖析

阿里云-云计算业务部

王立

(Email:edward.wang@alibaba-inc.com

Weibo:王立-阿里云**)**

目　录

第一章 引言

Amazon Web Services can be as big as the company's core e-commerce business.

--- **Jeff Bezos**, Founder & CEO of Amazon

自上世纪八十年代起，随着全球科技、文化和经济的发展，人类社会逐渐开始从工业化社会向信息化社会过渡；自九十年代中期起，经济全球化趋势推动信息技术高速发展，以 Internet 为代表的信息技术开始大规模应用于商业领域。在全球经济持续发展与增长的同时，企业信息化过程中暴露出来的问题亦逐渐凸现，如复杂的管理模式、失控的运营成本、困难的扩展支撑等等。由此应运而生的解决方案即是以便利低廉的服务方式提供从基础架构到软件应用、从数据到计算能力的各类资源，日益受到越来越多的用户，尤其是中小企业的欢迎。

2005 年 11 月 2 日，Amazon 正式发布其首个 Web 服务：Amazon Mechanical Turk[1]，白驹过隙、光阴荏苒，数年间，Amazon 陆续推出了 30 余种丰富多样的 Web 服务，拥有遍布 190 多个国家的数十万用户，预期 2015 年的营收有望达到 26 亿美元[2]，成为云计算业务事实上的执牛耳者。

本文将介绍 Amazon Web Services 12 个门类共 33 种云计算产品与服务，关注基于 AWS 的多种解决方案，最后浅谈了一下作者本人对 AWS 未来发展方向的想法。

因时间有限，加之作者本人水平有限，对 AWS 的理解存在偏差谬误，在所难免，敬请读者谅解并反馈修改意见，深表谢意！

第二章 计算类（Compute）

2.1 Amazon Elastic Compute Cloud (EC2)

2.1.1 简介

2006 年 8 月，Amazon 发布了 Amazon EC2[3] Beta 版，与之前刚刚推出的 Amazon S3 一起徐徐揭开了云计算业务时代的大幕，并逐步帮助 Amazon 确立了作为云计算基础服务提供商的领导地位。

EC2 提供了一种基于 Xen 的可信及可伸缩的虚拟计算环境，用户可根据业务需要租用不同配置的虚拟机，并在其上运行标准或自定义的镜像文件。EC2 支持动态且自动地 scale up/down 计算资源，获取与启动实例均在分钟级别，同时，EC2 还提供了弹性负载均衡机制。

EC2 上部署并发布服务的基本单元为 Amazon Machine Image (AMI，以 S3 对象方式存储)[4]，Amazon 及 EC2 社区提供了上千种预定义的镜像文件 AMI，用户可以在租用的虚拟机上运行这些模板化的镜像文件，也可以创建并运行包含自己应用、数据和相应配置的 AMI。

用户对所租用的 EC2 计算资源具有完全的控制权，包括安全配置、网络访问等。同时，EC2 还提供了 Web Service API 或管理工具协助用户管理及监控虚拟机实例、网络及存储服务等。

EC2 可与其它 AWS 有机结合起来使用，从而为用户带来更大的价值。如 EC2 可以和 Amazon S3[5]、Amazon Relational Database Service（RDS）[6]、Amazon SimpleDB[7]及 Amazon Simple Queue Service（SQS）[8]一起提供计算、查询处理和存储的完全解决方案；

早期的 EC2 版本中，用户数据存储在实例中，一旦实例被破坏，用户数据随之丢失。为了解决这一问题，EC2 推出了 Amazon Elastic Block Store（EBS）[9]，为 EC2 提供可靠且弹性的块存储服务。

- Amazon EBS 允许用户创建 1GB 到 1TB 的存储卷，存储卷类似没有初始化的原始设备，带有设备名及块设备接口。EC2 实例可以挂载存储卷，在其上创建文件系统，或者直接当作块设备使用；
- Amazon EBS 卷只能为同一 Availability Zone 中的 EC2 实例挂载，多个存储卷可以同时挂载到一个实例上，但一个卷只能同时挂载到一个实例上；存储卷会自动在 Availability Zone 内部执行副本复制操作；
- Amazon EBS 支持生成卷的快照 snapshot，并将其存储到 Amazon S3 上；第一份 snapshot 是全量的，之后均为增量快照；
- 用户可通过 Amazon CloudWatch[10] API 或 AWS 管理控制台[30]来访问 EBS 卷的性能指标，包括带宽、吞吐量、延迟及队列长度等。

可靠性必然是用户使用 EC2 所需考虑的一个重要方面，Amazon EC2 SLA 为每个 EC2 Region 承诺了 99.95%的可用性；安全是另一个不可忽视的因素，EC2 提供了多种机制来保护用户的计算资源，如可通过 Web Service 接口配置防火墙来控制实例之间的网络访问；同时，EC2 还提供了 Amazon Virtual Private Cloud（VPC）[11]来实现更为严格的安全配置。有

关 AWS 安全的更多信息请参考相关文档[12]；

EC2 Region 是地理上隔离的，处于不同的地区或国家，目前 EC2 有 8 个独立的 Region。每个 Region 包含一个或多个 Availability Zone，各个 Zone 之间相互独立。基于 Region 和 Availability Zone 的设计，EC2 实例可部署在多个位置以避免单点故障的影响。

2.1.2 服务亮点

EC2 具有如下亮点，现在已成为弹性计算服务的事实标准：

- 弹性：用户可在分钟级别增减 EC2 的容量，并同时操作多达上千个实例。所有这些操作都可基于 EC2 提供的 Web Service API 完成，因此用户的应用可以按需自动向上或向下扩展。
- 完全可控：用户对自己的 EC2 实例具有完全控制权。用户具有每个实例的 root 权限、可以使用 Web Service API 启停实例（包括远程重启）、还可访问实例的输出。
- 灵活性：用户可以选择多种实例类型、操作系统及软件包。用户可以自定义配置信息，包括内存、CPU、内置存储、根据所选操作系统及应用而优化的根分区大小等。例如，操作系统可选项包括多种 Linux 发布版本、Microsoft Windows Server 等。
- 与其它 AWS 集成：EC2 与 Amazon S3、Amazon RDS、Amazon SimpleDB 及 Amazon SQS 结合使用可提供跨大范围应用的计算、查询处理及存储的完全解决方案。
- 可靠性：EC2 提供了高可靠的计算环境，后备实例可迅速且预知性地替换故障实例。服务运行在 Amazon 可信网络基础架构及数据中心内，EC2 SLA（Service Level Agreement）[13]为每个 Amazon EC2 Region 承诺了 99.95%的可用性。
- 安全：EC2 提供了多种机制保证用户计算资源的安全性。
  - EC2 提供了 Web Service 接口来配置防火墙设置，以控制对实例组及实例组之间的网络访问；
  - 当在 Amazon VPC 内部启动 EC2 资源时，用户可以通过指定希望使用的 IP 范围来孤立自己的计算实例，并使用工业标准级别的加密 IPsec VPN 来连接到已有的 IT 基础架构。用户也可在 VPC 内部启动专有实例（Dedicated Instance）[14]，专有实例是在 VPC 中运行的 EC2 实例，且提供了额外的独立性以专供一个用户使用；
  - 更多 EC2 安全的信息请参考 AWS 安全文档[12]。
- 廉价：Amazon EC2 按用户实际使用计算资源的情况收取很低廉的费用[15]。
  - 按需实例（On-Demand Instance）：用户可按小时支付对计算资源的使用费用，从而大大降低了固定成本的投入，且省却了用户为应付网络周期性峰值而做的边界投入；
  - 保留实例（Reserved Instance）：用户可通过一次性支付（将在每小时费用上有个很大的折扣）来保留一个实例，保留实例有三种类型（轻量级、中型、重量级）供用户选择；
  - Spot 实例[16]：Spot 实例允许用户对未使用的 EC2 资源出价，只要出价超过 Spot 价格，就可以一直使用这些资源。Spot 价格基于供求关系而周期性变化，用户的出价匹配或超过 Spot 价格的可以获得对可用 Spot 实例的使用权。如果用户应用的运行时间可灵活变化，则使用 Spot 实例可以显著地降低 EC2 开支。
- 易于上手：用户可以访问 AWS Marketplace[17]，选择预配置好的软件 AMI，以快速开始使用 EC2，并可利用一键运行功能或 EC2 终端来快速部署该软件。

2.1.3 特点

Amazon EC2 提供了一组强大的功能来构建可扩展、错误弹性恢复的企业级应用，包括：

- Amazon EBS：Amazon EBS 为 EC2 实例提供了持久存储机制。EBS 卷是网络连接式的，其持久化独立于 EC2 实例的生命周期。EBS 卷具有高可用性、高可靠性，其可作为 EC2 实例的根分区或作为标准的块设备连接到某个运行中的 EC2 实例。当作为根分区使用时，EC2 实例可被停止，之后再重启，这样在用户保存实例状态的同时，也只需为所使用的存储资源支付费用。EBS 卷比 EC2 本地存储具有更高的可靠性，因为 EBS 卷会在后端（单个 AZ 中）自动备份。如果需要更高的可靠性，EBS 还可在某时间点为用户卷创建一致性快照，存储在 S3 上，并自动复制到多个 AZ 中。这些快照可以作为新的 EBS 卷起点，以长期可靠地保护用户数据。用户也可和合作者及其它 AWS 开发人员分享这些快照。EBS 提供了两种类型的卷：标准卷（Standard volume）与已配置的 IOPS（Input/Output operations Per Second）卷（Provisioned IOPS volume）。标准卷提供了成本低廉的存储，适合具有中等或突发 I/O 需求的应用；已配置的 IOPS 卷适合 I/O 密集型应用，如数据库等。
- EBS 优化的实例：通过支付很少的按小时计费的额外费用，用户可以“EBS 优化”的模式启动 EC2 实例，EBS 优化实例能使 EC2 实例充分利用部署在 EBS 卷上的 IOPS。EBS 优化实例在 EC2 和 EBS 之间提供了专门的吞吐率设置，依赖于所使用的实例类型，吞吐率处于 500Mbps 至 1000Mbps 之间。能以 EBS 优化实例方式启动的 EC2 实例类型参见[18]。
- 多个位置：EC2 实例可放于多个位置，EC2 的位置由 Region 和 AZ 组成。AZ 之间地点各异以实现故障隔离，同一 Region 中的 AZ 之间具有低价格、低延迟的网络连接。通过在不同的 AZ 中启动实例，可以避免用户应用的单点故障。Region 由一个或多个 AZ 组成，AZ 在地理上是分开的，处于不同的地理区域或国家中。EC2 目前有 8 个 Region：美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）、欧盟（爱尔兰）、亚太区（新加坡）、亚太区（东京）、南美洲（圣保罗）及 AWS GovCloud[19]。
- 弹性 IP 地址：弹性 IP 地址是为动态云计算而设计的静态 IP 地址。弹性 IP 地址与用户账号而非特定实例相关联，用户控制该地址的使用直到明确释放其为止。但与传统静态 IP 地址不同的是，弹性 IP 地址可通过编程方式将公共 IP 地址与用户备用实例重新映射，从而屏蔽用户对实例或 AZ 故障的感知。
- 虚拟私人云（VPC）：VPC 是在公司现有 IT 基础架构和 AWS 云之间的一个安全无缝的桥梁。VPC 使企业能够通过虚拟私人网络（Virtual Private Network，VPN）将其已有的基础架构与一组独立的 AWS 计算资源相连接，扩展其管理能力，如安全服务、防火墙、入侵检测等。
- Amazon CloudWatch：CloudWatch 提供了监控 AWS 云资源和应用的服务。用户通过 CloudWatch 可以查看资源利用、操作性能及总体的需求模式－包括 CPU 利用率、磁盘读写、网络流量等信息，数据将以统计资料、图表、报警提醒等方式显示，用户也可以提供自己业务或应用的度量数据。CloudWatch 的数据可通过 Web Service API 或命令行工具访问。
- 自动扩展（Auto Scaling）[20]：Auto Scaling 可使用户根据预定义的条件来自动向上或向下扩展 EC2 容量。借助 Auto Scaling，在请求高峰时，自动无缝地向上扩展 EC2 实例数目以保持性能；在请求波谷时，自动向下扩展以尽力降低开支。因此，

Auto Scaling 尤其适合使用量每小时、每日、每周变化的应用。Auto Scaling 基于 CloudWatch，无需支付额外的费用。

- 弹性负载均衡（Elastic Load Balancing，ELB）[21]：ELB 自动将应用入口流量分发到多个 EC2 实例上，使得应用具有更高的容错性。ELB 可以监测实例池中不正常的实例，并自动将流量转移到正常的实例上，直到不正常的实例恢复状态。ELB 可在单个 AZ 或跨 AZ 部署。Amazon CloudWatch 可用来监控 ELB 的运行情况，包括请求数、请求延迟等。一旦购买了 ELB，这些监控是免费提供的。
- 高性能计算（High Performance Computing，HPC）集群[22]：基于 EC2，对于复杂计算应用，如紧耦合并行处理，或对网络性能敏感的应用，用户一方面可以通过自建的基础架构来获得高计算和网络性能，同时又能具备 EC2 附带的弹性、灵活性和成本优势。集群计算与 GPU 实例通过特殊的调优，能提供高性能的网络容量，并可通过编程方式在集群中启动。这样，应用可获得低延迟的网络性能，以满足紧耦合与点对点通信的需求。集群计算与 GPU 实例还显著提升了吞吐率，使得它们非常适合网络密集操作型的应用。
- 高 I/O 实例：高 I/O 实例适合于超高速、低延迟、随机 I/O 访问的应用场景。高 I/O 实例属于 EC2 实例类型的一种，可提供超过 100,000 IOPS 的随机 I/O 速率。高 I/O 实例基于固态硬盘（Solid State Disk，SSD）技术，是超高性能 NoSQL 及关系型数据库应用的理想之选。
- VM 导入导出[23]：用户可轻松地从现有环境中将虚拟机映像文件导入 EC2 实例中，并在任何时候将其导出到用户自己的环境中。这项功能是免费提供的。
- AWS Marketplace[24]：AWS Marketplace 是一个在线商店，用户可以从中寻找、购买并快速部署运行在 AWS 上的软件。用户可使用 AWS Marketplace 的一键部署功能来快速启动预配置好的软件，并按小时或月来支付使用费用。

2.1.4 实例类型

Amazon EC2 实例具有以下类型：

1) 标准实例

标准实例包括一族实例类型，适合大部分应用的需求，如表 2-1 所示。

表 2-1 EC2 标准实例族

实例类型	内存	计算单元	本地实例存储	平台
小型实例（默认）	1.7 GB	1[1]	160 GB	32 位或 64 位
中等实例	3.75 GB	2[2]	410 GB	32 位或 64 位
大型实例	7.5 GB	4[3]	850 GB	64 位
超大型实例	15 GB	8[4]	1690 GB	64 位

2) 微型实例

微型实例具有 613 MB 内存，短期内可以提升到 2 个计算单元，只有 EBS 存储，

[1] 1 个虚拟核（Virtual Core）上运行 1 个 EC2 计算单元（EC2 Compute Unit，ECU）。1 个 ECU 相当于 1.0-1.2 GHz 2007 Opteron 或者 2007 Xeon 处理器。
[2] 1 个虚拟核上运行 2 个 EC2 计算单元。
[3] 2 个虚拟核上各运行 2 个 EC2 计算单元。
[4] 4 个虚拟核上各运行 2 个 EC2 计算单元。

32 位或 64 位平台。微型实例很适合一般情况吞吐率比较低的应用和网站，只周期性地短暂需要更多的计算资源。

3) 高内存实例

这一族实例类型为高吞吐率的应用提供大内存的支持，如数据库和内存缓存应用等，具体类型如表 2-2 所示。

表 2-2 EC2 高内存实例族

实例类型	内存	计算单元	本地实例存储	平台
超大内存	17.1 GB	6.5[1]	420 GB	64 位
双倍超大内存	34.2 GB	13[2]	850 GB	64 位
四倍超大内存	68.4 GB	26[3]	1690 GB	64 位

4) 高 CPU 实例

相较其它类型，这一族实例类型的 CPU 资源比例上要多于内存，特别适合计算密集型应用，具体类型如表 2-3 所示。

表 2-3 EC2 高 CPU 实例族

实例类型	内存	计算单元	本地实例存储	平台
中等实例	1.7 GB	5[4]	350 GB	32 位或 64 位
超大实例	7 GB	20[5]	1690 GB	64 位

5) 集群计算实例

这一族实例提供了更高的 CPU 资源和网络性能，尤其适合高性能计算应用以及其它对网络带宽流量有要求的应用，具体类型如表 2-4 所示。有关 HPC 的案例和集群管理选项请参看[22]。

表 2-4 EC2 集群计算实例族

实例类型	内存	计算单元	本地实例存储	平台	带宽
四倍超大	23 GB	33.5	1690 GB	64 位	10 Gigabit
八倍超大	60.5 GB	88	3370 GB	64 位	10 Gigabit

6) 集群 GPU 实例

这一类型实例提供了通用图像处理单元（Graphics Processing Unit，GPU）及高 CPU 和网络性能，适合高并行处理（包括 HPC）渲染和媒体处理应用。集群计算实例提供了创建一个低延迟、高吞吐率网络连接的实例集群的能力，而集群 GPU 实例提供了额外的选项，相较传统处理器，其可使应用从 GPU 的并行计算能力带来的效率提升中获益，具体配置如表 2-5 所示。这类实例类型可用于 HPC 应用中[22]。

[1] 2 个虚拟核上各运行 3.25 个 EC2 计算单元。
[2] 4 个虚拟核上各运行 3.25 个 EC2 计算单元。
[3] 8 个虚拟核上各运行 3.25 个 EC2 计算单元。
[4] 2 个虚拟核上各运行 2.5 个 EC2 计算单元。
[5] 8 个虚拟核上各运行 2.5 个 EC2 计算单元。

表 2-5　EC2 集群 GPU 实例

实例类型	内存	计算单元	GPU	本地存储	平台	网络带宽
四倍超大	22 GB	33.5	2 x NVIDIA Tesla “Fermi” M2050	1690 GB	64 位	10 Gb

7)　高 I/O 实例

这一类型的实例提供了非常高的磁盘 I/O 性能，非常适合高性能数据库负载的要求。高 I/O 实例采用基于 SSD 的本地存储，亦提供了高级别的 CPU、内存和网络性能配置，具体参数如表 2-6 所示。

表 2-6　EC2 高 I/O 实例

实例类型	内存	计算单元	本地实例存储	平台	带宽
四倍超大	60.5 GB	35	2*1024 GB SSD	64 位	10 Gb

2.1.5 定价

按用付费，根据实例类型以及所部署的 Region 不同，费用稍有差异。使用 EC2 时产生的 Internet 带宽出入流量亦要收取相应费用，但同一 Region AWS 内部（如 EC2 和 S3 之间）的数据传输则是免费的。此外，使用相关业务，如 EBS、CloudWatch 均会产生费用，具体收费标准请参考[3]。

2.2 Amazon Elastic MapReduce (Amazon EMR)

2.2.1 简介

Amazon EMR[25]使用 Hadoop 作为其分布式处理的引擎，通过在 EC2 & S3 上架构 Hadoop 框架来提供大数据处理服务，即在 EC2 实例集群上（如果超过 20 个实例，需要另提申请）运行 MR 任务，并将用户的处理程序、源数据及处理结果存储在 S3 上，也可选择保存在 Amazon DynamoDB 中。Amazon EMR 允许用户使用 Java/ C++/Perl/Ruby/Python/PHP/R 等语言编写自己的处理程序。

Amazon EMR 的应用非常简单，例如最常用的，用户可使用 Java 编写 Mapper/Reducer，将编译好的 Jar 包和源数据上传到 S3，然后通过 EMR CLI 或 API 启动一个 JAR 工作流来执行程序获得结果；也可以使用 Python、Ruby 或 PHP 等编写 Mapper/Reducer，将编好的程序和源数据上传到 S3，然后通过 EMR CLI 或 API 启动一个 Hadoop streaming 工作流来执行程序获得结果，结果也将被写到用户在 S3 上 bucket 中；又如，用户可以编写 Hive 脚本来创建一个 Hive 工作流处理数据。用户可通过 AWS 管理控制台来监控工作流的运行情况，工作流的状态信息保存在 Amazon SimpleDB 中。

2.2.2 特点

Amazon EMR 除了具有 AWS 所共有的弹性、可靠、低成本等特点外，与包括 Amazon EC2、S3、DynamoDB 在内的相关 AWS 也实现了无缝集成，同时，EMR 还集成了大量第三方工具，提供包括 SQL 查询、数据分析在内的诸多功能。

2.2.3 定价

按用付费，根据实例类型以及所部署的 Region 不同，费用稍有差异。具体收费标准请参考[25]。

2.3 Amazon Auto Scaling

2.3.1 简介

Auto Scaling[20]与 Amazon EC2 配合使用，它允许用户自定义条件以根据业务的性能情况自动无缝地增加/减少所租用的 Amazon EC2 实例。Auto Scaling 是基于 Amazon CloudWatch 的计量结果来实现的。Amazon EC2 用户可以直接通过 API 或命令行工具使用 Auto Scaling 服务。

为了使用 Auto Scaling，用户需先下载 Auto Scaling 及 Amazon CloudWatch 的命令行工具，创建一个组（Auto Scaling Group），其中包含所有想 Auto Scaling 的 EC2 实例；之后定义启动一个新的 EC2 实例所需使用的配置参数，最后设定策略，指定在什么条件下该采取什么样的 scaling 操作。常见的策略用法如：

- 自动 scale Amazon EC2 实例
  用户可自定义条件为：如果 EC2 实例集群的 CPU 利用率上升超过某个值，则加入若干新的 EC2 实例到集群中，反之亦然。
- 维持 Amazon EC2 实例集群的数量稳定
  用户也可自定义条件为：EC2 集群必须有多少个实例，如果其中任何一个实例因意外而停止工作，则添加一个新的实例到集群中。

2.3.2 定价

Auto Scaling 基于 Amazon CloudWatch，所以不另收取费用。

2.4 Amazon Elastic Load Balancing

2.4.1 简介

Amazon Elastic Load Balancing[21]可以在一个或多个 Availability Zone 中的 Amazon EC2 实例之间自动分配入口流量。Elastic Load Balancing 可以检测 EC2 实例的健康情况，一旦发现某个实例有问题，即可在其恢复正常之前自动地将流量重定向到其它健康的实例上。

Amazon EC2 用户可以直接使用 Elastic Load Balancing，按照 AWS 管理控制台（AWS Management Console）的配置向导即可方便快捷地完成负载均衡器及其之后 EC2 实例的设置，亦可通过 Elastic Load Balancing API 和命令行操作完成上述操作[26]，此处从略。

2.4.2 特点

ELB 具有以下特点：

- ELB 可在处于一个或多个 AZ 中的 EC2 实例之间分布入口流量，并随应用入口流量的变化而自动扩展其请求处理能力；
- 当用于 VPC 中时，用户可以创建并管理与 ELB 相关联的安全组，来提供额外的网络和安全选项；
- 当用于 VPC 中时，用户可以创建一个无公共 IP 地址的内部负载均衡器（不面向 Internet）；
- ELB 可以监测用于负载均衡的 EC2 实例的健康状况，一旦发现不正常的 EC2 实例，就不再将流量分发到该实例，而是在其它正常 EC2 实例之间分配；
- ELB 支持将用户 session 和某个 EC2 实例绑定；
- ELB 在负载均衡器上支持 SSL termination，包括从应用实例 offload SSL 解密、集中管理 SSL 认证、使用可选的公开密钥认证加密至后台实例等；
- 灵活的密码支持，用户可控制用于 SSL 协商客户端连接时 ELB 所接受的密码和协议；
- ELB 支持 IPv4 和 IPv6；
- 包括请求数、请求延迟在内的 ELB 计量参数由 Amazon CloudWatch 监控显示。

2.4.3 常用场景

Amazon ELB 具有以下几个方面的特点，用户可以利用这些特点架构更灵活强壮的应用：

- 实现更佳的容错性能
  为了达到容错的目的，用户可将自己的 Amazon EC2 实例放置于多个 AZ。而为了得到更佳的容错性能，可在计算实例前端部署 Amazon ELB。ELB 将自动实现跨实例和 AZ 的流量均衡，且确保只有健康的 EC2 实例来处理服务。用户可跨 EC2 实例（在一个 AZ 或多个 AZ 中）部署 ELB 以均衡应用的入口流量。ELB 将自动监测 EC2 实例的健康状况，一旦发现某个 EC2 实例出现故障，就不会再向其分发流量。如果用户在多个 AZ 中部署了 EC2 实例，而处于某个 AZ 中的所有 EC2 实例均出现问题，则 ELB 将把流量均衡到其它 AZ 中的 EC2 实例上。当故障 EC2 实例恢复正常后，ELB 将纳入这些节点继续负载均衡。
- 自动扩展（Auto Scaling）

设想用户有这样一个需求：期望在 ELB 之后的健康 EC2 实例不能少于 2 个。我们可以采用 Amazon Auto Scaling 设置该条件，一旦 Auto Scaling 发现健康实例不足 2 个，将自动添加必需数目的 EC2 实例到 Auto Scaling 组中。或者用户也可以添加这样的 Auto Scaling 条件：即要求在任何一个 15 分钟周期内，一旦某个 EC2 实例的延迟超过 4 秒就要添加新的 EC2 实例。由此可见，Auto Scaling 可以和 ELB 无缝集成。

- 在 Amazon VPC 中使用 ELB
  ELB 处于 VPC 和 Internet 之间，可为用户 VPC 轻松创建 Internet 入口点，并可在 VPC 内部为应用的不同层级间提供负载均衡。用户可为 ELB 指定安全组，以控制哪个端口允许哪些源的访问。因为 ELB 依附于 VPC 之上，故所有现有的网络访问控制列表（Network Access Control Lists，ACL 形式）和路由表（Routing Tables）都能继续正常地提供额外的网络控制能力。用户在 VPC 内创建负载均衡器后，可指定该均衡器是面向 Internet（默认）还是面向内部的。如果选择面向内部，就不需要在负载均衡器之前设置 Internet 网关，且负载均衡器的 DNS 记录中将使用其私有 IP 地址。

2.4.4 定价

Elastic Load Balancing 按其运行时长（不足一小时按一小时计，0.025 美元/hour）及通过负载均衡器的流量（0.008 美元/GB），月末结账。

第三章 内容分发类（Content Delivery）

3.1 Amazon CloudFront

3.1.1 简介

Amazon CloudFront[27]是用于内容分发的 Web Service，与其它 AWS 一起提供了一种简便的方式，使开发者和商业应用能够低延迟、高数据传输率地将内容分发给终端用户。Amazon CloudFront 可使用遍布全球的节点服务器（参见 3.1.5 节）来分发整个网站，包括动态、静态和流内容。对内容的请求将被自动路由到离用户最近的节点服务器，因此，内容的分发可能具有最优的性能。经过优化，Amazon CloudFront 可与 Amazon S3、EC2、ELB 及 Amazon Route 53[28]协同工作，Amazon CloudFront 也可和任何存储有用户原始文件的非 AWS 的源服务器无缝集成。

3.1.2 功能

Amazon CloudFront 的 Web Service 接口简单易用，内容被组织在分发包（distribution）中。分发包中指定了文件原始版本的位置，每个分发包都有一个唯一的 CloudFront.net 域名（例如，abc123.cloudfront.net），用户可在全球节点服务器中用该域名来指代分发的内容，使用者也可将自己的域名（例如，www.example.com）映射到分发包。创建分发包之后，既可使用 HTTP 或 HTTPS 协议来下载内容，也可使用 RTMP（Real Time Messaging Protocol，实时消息传输协议）协议将内容转化为流。CloudFront 的可用性在 CloudFront SLA 中定义[29]。

Amazon CloudFront 的使用步骤如下：

1) 将原始版本的文件存储在一个或多个源服务器上，源服务器即为文件最后版本的存储位置。S3 bucket、EC2 实例、ELB 或自有服务器都可用来作为源服务器。
2) 创建一个分发包，并通过一条 API 调用或 AWS 管理端将源服务器注册到 CloudFront 中。如果源服务器有多台，使用 URL 模式匹配来指定每台源服务器上分别存储有什么内容。可指定一台源服务器为默认源。
3) 终端用户可在网页、媒体播放器或应用中使用分发包的域名来请求某个对象，请求将自动被路由到离用户最近的节点服务器，以保证高效的内容分发。
4) CloudFront 按每月产生的数据传输量和请求数计费。

3.1.3 特点

Amazon CloudFront 具有以下特点：

- 快速：CloudFront 在全球部署有节点服务器，通过缓存静态内容的副本，使内容能够以低延迟、高且持续的数据传输率进行分发。对动态内容的请求将通过优化的网络路径传回给源服务器（运行在 AWS 中，如 EC2，ELB），以保证更可靠一致的

用户体验。Amazon 负责持续监控这些网络路径，从 CloudFront 节点服务器到源服务器之间的连接将被重用于动态内容分发，以尽力获取最佳的性能。

- 简单：仅需一条 API 调用，就可通过 CloudFront 网络从 S3 bucket、EC2 实例或其它源服务器分发内容，也可以使用 AWS 管理控制台的 GUI 与 CloudFront 交互。用户可使用同一域名指向网站的所有内容，无需为静态和动态内容分别创建域名。对现有配置的任何修改都可在几分钟内在整个全球网络生效。
- 与其它 AWS 天生兼容：CloudFront 可使用 Amazon S3 来永久存储静态文件的最终版本，使用 EC2 来生成动态内容，还可将 Web 应用部署在 ELB 之后的 EC2 服务器上，来分发全部网站内容。
- 成本经济：用户只需为内容分发量和请求数付费。分发内容包括静态、动态、流媒体及 Web 应用等。
- 弹性：CloudFront 会自动应对请求数增减的情况，无需用户人工干预。CloudFront 在节点服务器采用了多级缓存机制，并把对同一对象的并发请求归并折叠后再发给源服务器，这些优化进一步降低了扩展源服务器基础架构的需求。
- 可靠性：CloudFront 基于 Amazon 的高可靠性基础架构。CloudFront 会根据网络情况自动将用户请求路由到最近的可用节点服务器上，从节点服务器到 AWS 源服务器（例如，EC2、S3 等）之间的请求跑在 Amazon 监控的网络路径上，并从可用性和性能上进行了优化。
- 全球化：CloudFront 拥有遍布全球的节点服务器，包括美国、欧洲、亚洲和南美洲。

3.1.4 常用场景

Amazon CloudFront 有很多非常不错的应用场景，包括：

- 分发全部网站或 Web 应用

  一个典型的网站通常包含静态和动态内容，CloudFront 通过以下方式来提高网站分发的性能：

  - CloudFront 可在每个节点服务器缓存静态内容，包括图片、样式表单、JavaScript 代码等，浏览器可以从最近的节点服务器下载静态内容，而无需向源服务器发送请求，这样可降低延迟。
  - 终端用户请求动态或交互式内容时，最近的节点服务器会处理请求，并传回给运行在 AWS Region 中的源服务器，回传路径经过性能优化。CloudFront 还会重用节点服务器和源服务器之间的连接，以减少请求的连接建立延迟。CloudFront 还采取了其它连接优化措施来避免网络瓶颈，并充分利用节点服务器和用户之间的可用带宽。
  - 可用一个 CloudFront 分发包（故只需一个域名）分发全部网站，包括静态、动态和交互式内容，但可以用不同的源服务器存储不同类型的内容。CloudFront 提供了更精细化的控制，为网站中不同的 URL 配置不同源服务器和缓存参数。

- 分发软件或其它大文件

  CloudFront 具有很高的数据传输率，适合软件开发人员将所开发的软件、更新等分发给终端用户。虽然这些文件也可以放在 S3 上供终端用户下载，但 CloudFront 具有更高的应用层次和更低廉的价格。

- 分发媒体文件

  如果应用包含经常被访问的富媒体（rich media）文件，如音频或视频文件，采用

CloudFront 可以降低数据传输价格，并提高数据传输速度。CloudFront 为分发录播和实时媒体文件提供了多种选择：

- 录播媒体流：将原始 Adobe RTMP 流媒体文件存储在 S3 上，并使用 CloudFront 分发。可通过一条 API 调用或 AWS 管理控制台的简单操作来配置媒体流。
- Progressive download：将媒体内容的原始版本存储在 S3 上，配置 CloudFront 以 progressive download 方式下载视频和音频文件的分发包，并在节点服务器上缓存热门媒体文件。
- 实况分发：如果要向全球观众分发实况节目的音频或视频，CloudFront 可以缓存一小段时间内的实时媒体文件，并把对同一小段媒体文件的并发请求合并后在发给源服务器，这样可以大大降低对源服务器架构的请求压力，提高性能。此外，CloudFront 的实时 HTTP 解决方案可以帮助用户采用不同的设备平台（基于 Flash 或 Apple iOS 设备）来分发实况节目。

3.1.5 产品细节

- Amazon CloudFront 全球节点服务器网络
  Amazon CloudFront 全球节点服务器分布如表 3-1 所示：

表 3-1 Amazon CloudFront 节点服务器分布

美国	弗吉尼亚州 Ashburn (2 台)、得克萨斯州达拉斯/沃斯堡（2 台）、佛罗里达州 Jacksonville、佛罗里达州迈阿密、纽约州纽约（2 台）、新泽西州 Newark、加利福尼亚州洛杉矶（2 台）、加利福尼亚州 Palo Alto、加利福尼亚州 San Jose、华盛顿州西雅图、印第安纳州 South Bend、密苏里州圣路易斯
欧洲	阿姆斯特丹（2 台）、都柏林、法兰克福（2 台）、伦敦（2 台）、米兰、巴黎（2 台）、斯德哥尔摩
亚洲	香港、大阪、新加坡（2 台）、悉尼、东京
南美洲	圣保罗

- Amazon CloudFront 分发包
  分发包有两种类型：HTTP/HTTPS 下载分发包和 RTMP 流分发包，每个分发包对应一个可用于 Web 应用中的唯一域名。
- HTTP 下载分发包
  我们可以为每个分发包配置 URL 匹配规则，CloudFront 通过这些规则来识别源服务器，例如可以定义这样的规则，所有匹配/images/*的请求以 S3 作为源，所有匹配*.php 的请求使用 EC2 实例作为源。使用下载分发包的步骤为：
  1) 将文件的原始版本存储在源服务器上；
  2) 调用 *CreateDistribution* API 创建分发包，返回新分发包的域名；
  3) 在网站或 Web 应用中使用第 2 步中的域名来创建对象的链接地址。
  当终端用户使用域名请求某个网页或内容时，CloudFront 选择一个最佳的节点服务器 A 处理请求。如果节点服务器 A 上没有所请求文件的副本，则 CloudFront 从源服务器上拷贝一份到该节点服务器 A 上。
  默认情况下，分发包可按 HTTP 或 HTTPS 协议分发，但也可配置分发包仅支持其中一种，如果 CloudFront 需要从源服务器获取文件时，它将使用与之相同的协议。

CloudFront 下载分发包具有如下特性：

- 缓存行为：缓存行为是为指定的 URL 模式而设置的一组规则，这些规则基于文件扩展名、文件名、或者 URL 路径的任何部分（例如，*.jpg）。下载分发包可配置多种缓存行为，CloudFront 将入口请求和 URL 模式列表进行匹配，如果有相匹配的，CloudFront 将执行该 URL 模式设置的缓存行为。每个缓存行为都可包含如下 CloudFront 配置值：源服务器名称、用户请求连接所用协议、最短过期时间、查询字符串参数及私有内容的可信签名者。
- 源服务器：CloudFront 下载分发包可配置一个或多个源服务器[31]，源服务器既可是 AWS 资源（如 S3、EC2、ELB），也可是非 AWS 的自有服务器。因为 CloudFront 通过匹配 URL 来从不同的源服务器请求内容，故可充分利用 AWS 资源的特性，如 S3 可用来做存储源服务器、EC2 用来做计算源服务器等。使用自有服务器时，无需迁移数据或重新部署应用。
- 终端用户连接协议：默认支持 HTTP 或 HTTPS 协议，也可配置予以限定。
- 最短过期时间：可在缓存控制头中设置文件的过期时间，CloudFront 使用过期时间来决定是否需要从源服务器更新文件。如果文件经常变化，就可以为文件设定一个较短的过期时间。CloudFront 接受的最短过期时间是 0 秒，即对每个请求，CloudFront 都会访问源服务器。CloudFront 也可执行特定的缓存控制指令，如 private、no-store 等，这常用于分发动态内容而无需在节点服务器缓存时。每个定义好的缓存行为都可设置唯一的最短过期时间，如果源对象没有设置缓存控制头，CloudFront 默认的过期时间为 24 小时。
- 查询字符串参数：通常用来返回由运行在源服务器上的脚本生成的定制内容。CloudFront 默认不会将查询字符串参数（例如，”?x=1&y=2”）传给源服务器，URL 的查询字符串部分在缓存中对象识别阶段就会被丢弃。但可配置将查询字符串作为缓存对象唯一标识符的一部分传给源服务器。这样，就可为不同的浏览者提供定制的网页，同时保留节点服务器缓存带来的性能优势。
- 默认根对象：可指定某个文件（例如，index.html）作为分发包的根对象，例如，http://abc123.cloudfront.net/ 对应到 index.html 文件。
- 对象版本和缓存失效：有两种方式来更新 CloudFront 节点服务器上的缓存文件。第一种方式采用对象版本来管理文件内容的更改。为实现对象版本机制，首先要在源服务器为文件的每个版本创建一个唯一的文件名，然后网页或应用中的文件根据其版本使用相应的文件名。这样，CloudFront 可以缓存任意版本的对象，而不必等到该对象过期后才能提供一个新版本。第二种方式是调用失效 API 来移除 CloudFront 所有节点服务器上的某个文件副本，而不论源服务器上该文件所设置的过期时间。如果需要同时删除多个文件，可在一个 XML 文档中指定文件列表（最多 1000 个）。缓存失效的特性被设计用于意外场景中，例如，矫正上传的视频文件的编码错误或网站 CSS 文件的意外升级等。但如果事先知道文件会经常更改，建议采用对象版本方式来管理文件的更新，因其可以控制何时让更改起作用，也可避免让对象失效带来的潜在费用开支。
- 访问日志：如果需要，可以打开 CloudFront 的访问日志功能，获取分发包的更多流量信息。访问日志是活动记录，显示了每个对内容的请求的详细信息。要使用访问日志功能，首先必须注册 Amazon S3 服务，然后创建或指定一个 Amazon S3 bucket 来存储访问日志。使用该功能没有额外的费用，但读写、存储日志会产生 S3 的费用。

- 按需（On-Demand）媒体文件的流分发包（streaming distribution）

Amazon CloudFront 允许用户创建“流分发包”来分发富媒体内容，这和 CloudFront 分发包是不一样的方式，流分发包将内容实时传给终端用户收看。流分发包采用 RTMP 协议及其若干变种，而不同于其它 CloudFront 分发包使用 HTTP 或 HTTPS 协议。CloudFront 使用 Adobe Flash Media Server 3.5 启动流分发包。

采用流有一些潜在的好处，能在回放时提供更多灵活性，比如很容易暂停、倒进、快进媒体文件到任何需要的位置，而无需考虑媒体文件有多少部分下载到浏览器了。我们还可配置流分发包来使用动态的位速流，一旦配置完成，即可存储同一视频的多个副本，每个副本以不同的质量标准编码。之后，流分发包将根据终端用户的网络连接速率，自动调整视频的画面质量。

采用流可以对内容有更多控制，因为终端用户看完视频后，不会在其计算器上留下任何文件。此外，流还可以节省开支，因为其仅仅分发被终端用户真正观看的那些部分的媒体文件。与此不同，传统下载通常将整个媒体文件下载下来，哪怕终端用户仅仅观看其中一部分。

流分发包支持很多种可被 Flash 播放的文件格式，其中包含流行的 FLV 和 MP4 文件格式以及 VP6 与 H.264 视频压缩标准。

设置完流分发包后，可使用 Amazon 的视频流诊断客户端[32]来测试视频。

- 实时媒体的 HTTP 流

  Amazon CloudFront 提供了两种方式来基于 HTTP（使用 Amazon CloudFront 下载分发包）分发实时媒体：

  - 使用 Adobe Flash Media Server 的实时流：CloudFront 可与运行着 Adobe Flash Media Server（FMS）的 EC2 结合起来，把实时 HTTP 流发送到 Flash Player 和 Apple iOS 设备。EC2（其上运行着 FMS）必须配置为 CloudFront 下载分发包的源。创建一个 AWS CloudFormation 模板[33]就可以处理实时流所需的全部部署和后续工作。采用 CloudFront 来搭建实时 HTTP 流的详细说明请参见[33]。
  - 使用 Windows Media Services 的实时平滑流：平滑流是微软可适应性流技术，将实时流传送到 Microsoft Silverligh 客户端。可使用 CloudFront 与运行着 Windows Media Services 的 EC2 结合起来提供实时平滑流，也可采用这种方式将实时流以 Apple HTTP Live Streaming（HLS）格式分发到 Apple iOS 设备。用户可创建 AWS CloudFormation 模板来自动部署实时流所需的 AWS 资源栈[34]。这样用户可以对源服务器（运行着 Windows Media Services 的 EC2 实例）具有完全控制权，从而可配置额外的 IIS 实时平滑流功能。

- Amazon CloudFront 与 Amazon Route 53 结合使用

  Amazon Route 53 是 AWS 的 Domain Name System（DNS）Web Service。与 CloudFront 类似，通过遍布全球的 DNS 服务器，Route 53 可以快速、低延迟地应答 DNS 查询。Route 53 使用 CNAME 记录将域名映射到 CloudFront 分发包。CNAME 允许 CloudFront URL 使用自有的域名，而非 abc123.cloudfront.net 类型的域名。

  使用 AWS 管理控制台可以配置和管理 Route 53 的 DNS 记录，这使得设置、更新 CNAME 记录变得更为容易。

- Amazon CloudFront 与 Amazon S3 结合使用

  S3 作为 CloudFront 源服务器来存储静态文件的原始版本具有更好的性能。我们知道，对经常被访问的静态文件（热门对象），可将其缓存到 CloudFront 的节点服务器以缩短用户的访问延迟，并降低费用开支。但当节点服务器的空间不足时，CloudFront 将删除较不热门的对象，以释放空间给更热门的对象。因此，对较不热门的对象，将要先访问 S3 而非直接从 CloudFront 上分发出来。S3 提供了很强的分

发性能，这样，即使需要持续拷贝较不热门的对象到节点服务器，也不会产生太多费用。

- Amazon CloudFront 与 Amazon EC2 和 EBL 结合使用

  使用 EC2 作为 CloudFront 源服务器的好处在于，可以使用相同的工具集配置并管理全部 Web 应用的分发，而且 CloudFront 节点服务器和 EC2 数据中心之间的路径是经过性能优化的，并一直处于监控之下，因此任何的网络路径问题都会被迅速监测到，并自动路由到其它正常的路径上，从而最小化对用户的影响。

  当运行多个 EC2 实例时，可使用 ELB 来自动分发来自于 CloudFront 节点服务器的应用入口流量。采用 ELB，可以在源服务器端获得更高的容错能力，提供 Web 应用的整体可用性。ELB 可部署在单个上或多个 AZ 间。

  如果想获取更高的可用性和源服务器连接性能，可以将应用实例跨多个 AWS Region 运行，每个 Region 部署一个 ELB endpoint。这样，就可以使用 Route 53 Latency Based Routing（LBR）特性将 CloudFront 源服务器请求路由到 AWS Region，以尽力降低到发出请求的 CloudFront 节点服务器的延迟。Route 53 与 CloudFront 集成在一起，从每个 CloudFront 节点服务器收集延迟测量数据，来进一步优化从源服务器读取数据的性能。

- 按使用收费

  同样的，CloudFront 仅按使用情况收费，这包括：

  - 源服务器费用：如果使用 S3 作为源服务器，需要支持 S3 的存储费用；如果使用 EC2 作为源服务器，需要支持 EC2 的计算费用。
  - 拷贝对象到节点服务器：如果请求对象不在节点服务器上，则节点服务器发送标准 GET 请求到源服务器。如果源服务器是 S3 或 EC2，将产生相应的 S3 或 EC2 的费用。
  - 节点服务器费用：HTTP/HTTPS 请求和数据出口费用。
  - 节点服务器对象失效费用：每月可免费失效最多 1000 个对象，之后收取费用（每个对象 0.005 美元）。

- CloudFront 的限制

  默认情况下，分发包支持的数据传输速率峰值为 1000 mbps，请求速率峰值为每秒 1000 个请求，超过此限制需要额外申请[35]。

3.1.6 定价

CloudFront 同样按使用情况收费，可使用 AWS 计算器[36]来预估每月的费用。CloudFront 的计费价格因地理位置以及所使用的节点服务器不同而有所不同[27]。

第四章 数据库类（Database）

4.1 Amazon Relational Database Services (RDS)

4.1.1 简介

Amazon 关系型数据库服务（Relational Database Service，RDS）[6]是一个 Web Service，使得云环境下建立、操作、扩展关系型数据库变得容易，在管理耗时的数据库任务的同时，提供了成本经济和容量可变的能力，使用户可以关注自己的应用和业务本身。

Amazon RDS 提供了访问 MySQL、Oracle 或 Microsoft SQL Server 数据库引擎的能力，这意味着基于现有数据库的代码、应用和工具可以用于 RDS 之上。RDS 自动为数据库软件打补丁、备份用户数据库、存储备份文件，并可在适当时候恢复数据库。用户可使用一个 API 调用来扩展与关系型数据库实例相关联的计算资源或存储容量，而且用备份来加强产品数据库的可用性和可靠性也更为容易。

4.1.2 功能

Amazon RDS 适合需要关系型数据库全部特性与能力的开发者或业务，也适合希望用关系型数据库来迁移现有的应用和工具的情况。通过 RDS，可以访问运行在 Amazon RDS 数据库实例之上的 MySQL、Oracle 或 Microsoft SQL Server 数据库引擎。

RDS 的使用非常简单：

- 使用 AWS 管理控制台或 RDS API 启动一个数据库实例（DB Instance），选择适合自己的 DB 引擎（MySQL、Oracle 或 SQL Server）、License 类型、DB Instance 类以及存储空间。
- 使用习惯的数据库工具或编程语言连接上一步中创建的 DB 实例，因为这是直连到 native 的 MySQL、Oracle 或 SQL Server 数据库引擎，故大部分为这些引擎设计的工具都可无需修改的运行在 RDS 上。
- 使用 Amazon CloudWatch（AWS 管理控制台或 CloudWatch API）来监控 DB 实例的计算和存储资源使用率，任何时候需要额外的容量时，可在控制台或通过简单的 API 调用来扩展与 DB 实例相关联的计算和存储资源。
- 用户仅需为实际使用的资源付费，包括 DB 实例的运行小时、数据库存储、备份存储及数据传输等。

4.1.3 服务亮点

Amazon RDS 具有以下服务亮点：

- 易于部署：通过 AWS 管理控制台或 API 简单调用，可在分钟级别完成适合产品使用的关系型数据库准备工作，无需担心数据库基础架构的部署、安装和维护工作。

- 管理：RDS 负责处理耗时的数据库管理任务，例如备份、补丁管理、备份等，用户可以关注应用和业务本身。
- 兼容性：RDS native 访问关系型数据库的方式使现有的工具和应用能与之兼容，此外，RDS 通过 DB 引擎版本管理（DB Engine Version Management）为用户的 DB 实例额外提供了控制 MySQL[37]、Oracle[38]的能力。
- 扩展性：使用 API 调用或 AWS 管理控制台可以扩展数据库的计算和存储资源，对 MySQL DB 引擎，还可为 DB 实例的部署关联一个或多个读副本（Read Replica）[39]，以应对 read-heavy 的数据库负载。
- 可靠性：RDS 有多个特点来加强关键产品数据库的可靠性，包括自动备份、DB 快照（snapshot）、自动替换主机、MySQL 和 Oracle 数据库引擎的多 AZ 部署等。
- 与其它 AWS 集成：RDS 与其它 AWS 紧密集成，例如，EC2 上应用采用同一 Region 中的 RDS DB 实例，可以获得低延迟的数据库访问。
- 安全：RDS 为 DB 实例提供了一组安全机制：
  - RDS 的 Web Service 接口可以配置防火墙设置来控制用户数据库的网络访问；
  - RDS 允许在 Amazon VPC 中运行 DB 实例，VPC 可以指定允许访问 DB 实例的 IP 地址范围，并通过工业级加密的 IPsec VPN 连接到用户现有的 IT 基础架构上。VPC 目前仅可用于 MySQL DB 引擎，且 RDS 微实例暂不支持 VPC[40]。
- 廉价：按需付费模式
  - 按需 DB 实例：按小时计费计算资源的使用；
  - 保留 DB 实例：有三种保留 DB 实例类型（轻型、中级、重量级），较之按需付费方式，可降低 30%-50%的开支[41]。

4.1.4 特点

Amazon RDS 提供的特性取决于所选择的 DB 引擎：MySQL 引擎[42]、Oracle 数据库引擎[43]、SQL Server 引擎[44]。

- 预设置的参数：RDS DB 实例预设置了一组参数，用户也可通过 DB 参数组（DB Parameter Groups）来额外控制。
- 监控与计量：RDS 可免费使用 CloudWatch 的计量功能，可通过 AWS 管理控制台查看 DB 实例部署中的关键操作计量参数，包括计算、内存、存储使用率、I/O 及 DB 实例连接数等。
- 软件自动打补丁：RDS 自动为用户所选的关系型数据库软件打补丁，用户也可使用 DB 引擎版本管理（DB Engine Version Management）来额外控制打补丁过程。
- 自动备份：RDS 可以在任意时间点备份、恢复 DB 实例。RDS 可按用户指定时段备份数据库及处理日志，并可在该时段内任一时间点恢复 DB 实例。自动备份时段最长为 35 天。
- DB 快照：DB 快照为用户触发的 DB 实例备份，可在任意时刻从快照恢复一个新的 DB 实例。
- 资源扩展：使用 RDS API 或在 AWS 管理控制台上简单几步操作，就可在数分钟内向上或向下扩展计算和内存资源。对 MySQL 和 Oracle 数据库引擎，可以在不宕机的情况下增加存储资源。
- 自动主机替换：当主机出现硬件故障时，RDS 将自动替换。
- 副本：RDS 提供了两种不同但互补的副本机制（目前仅支持 MySQL 数据库引擎）：多 AZ 部署[45]及读副本（Read Replica）[39]。这两种副本机制结合使用，可以增强

数据库的可用性，避免因意外的故障而丢失最新的数据库更新等。

- 隔离与安全：用户可将 DB 实例隔离在 VPC 中，并使用工业级标准的加密 IPsec VPN 连接到自己的 IT 架构上（目前仅支持 MySQL DB 引擎）[40]。此外，RDS 还可配置防火墙设置，并控制 DB 实例的网络连接。

4.1.5 DB 实例类

RDS 目前支持的 DB 实例类型如表 4-1 所示。

表 4-1　RDS 支持的 DB 实例类型

实例类型	内存	ECU	平台	I/O
微型	630 MB	最多 2 个（应对短而周期性峰值）	64 位	低（支持 MysQL 和 SQL Server）
小型	1.7 GB	1（1 虚拟核）	64 位	中等
大型	7.5 GB	4（2 虚拟核上各运行 2 ECU）	64 位	高
极大	15 GB	8（4 虚拟核上各运行 2 ECU）	64 位	高（仅支持 MySQL）
高内存极大	17.1 GB	6.5（2 虚拟核上各运行 3.25 ECU）	64 位	高
高内存双倍极大	34 GB	13（4 虚拟核上各运行 3.25 ECU）	64 位	高
高内存四倍极大	68 GB	26（8 虚拟核上各运行 3.25 ECU）	64 位	高

对每一种实例类型，RDS 提供 5GB 到 1TB 的存储空间选择。1 个 ECU 的能力相当于一颗 1.0-1.2 GHz 的 2007 Opteron 或 2007 Xeon 处理器。

4.1.6 定价

RDS 的价格与用户所选择的 DB 引擎、DB 实例类型、存储及数据传输的情况相关，详情请参见[6]。

4.2 Amazon DynamoDB

4.2.1 简介

Amazon DynamoDB[46]为完全受管的 NoSQL 数据库服务，提供了快速且可预期的性能与无缝扩展的能力。用户可通过 AWS 管理控制台启动 DynamoDB 数据库表，向上或向下热扩展表所需的资源，并可清晰地查看资源使用率与性能情况。DynamoDB 将数据存储在 SSD 上，并自动将表中数据和访问流量分布到多个服务器上，因此可以存储任意数量的数据并应对各级别的请求流量。DynamoDB 的数据会被自动备份到同一 Region 中的三个 AZ 上，因而具有天生的数据高可用性和持久性。

可使用 DynamoDB API[47]或 AWS 管理控制台来创建表并指定数据请求容量，使用 API 来读写数据，使用 AWS 管理控制台中的 CloudWatch 来监控 DynamoDB 表的状态与性能。

4.2.2 特点

- 扩展性：Amazon DynamoDB 具有无缝扩展吞吐率和存储的能力。在创建或更新表时，可以指定为表保留的读写容量大小，DynamoDB 据此保留必须的硬件资源以满足用户对吞吐率的一致性和低延迟要求。一个读写容量单位表示用户每秒一次读写操作（读写数据最大 1KB）。DynamoDB 的数据存储无大小限制，系统可以自动将一张表水平扩展到数百台服务器上。
- 快速且可预期的性能：因 DynamoDB 服务运行在 SSD 上，故 DynamoDB 服务端的平均延迟一般为几毫秒。
- 易于管理：用户只需创建数据库表，DynamoDB 会处理所有管理的任务。
- 天生的容错性：DynamoDB 会自动并同步用户数据备份到三个 AZ 上。
- 灵活性：DynamoDB 没有固定的 schema，每个数据项（item）都可以有数目不一的属性。
- 强一致性及原子计数器：与很多 NoSQL 数据库不一样的是，DynamoDB 更易于用户进行强一致性的读操作，以保证每次都能读取到最新的数据。DynamoDB 内置提供了原子计数器，用户可以通过 API 调用来获得增减数值的原子性操作。
- 成本经济：免费套餐允许每月 4000 万的数据库操作，收费套餐按每小时使用情况收取。
- 安全：DynamoDB 使用可靠的加密方法来认证用户，防止非授权的数据访问，并与 AWS 身份与访问管理服务（AWS Identity and Access Management，IAM）集成以提供更精细化的访问控制。
- 集成监控：AWS 管理控制台可以监控 DynamoDB 中表的关键操作，并与 CloudWatch 集成来展现数据库请求吞吐率、表延迟及资源消耗情况。
- 弹性 MapReduce 集成：DynamoDB 与 Amazon 弹性 MapReduce 服务（Amazon Elastic MapReduce，EMR）集成，使用户能对 DynamoDB 中的大数据集进行复杂的分析任务，并将结果存储在 Amazon S3 上，EMR 也可联合多个源（例如，DynamoDB、RDS、S3）的数据集进行复杂的分析任务，并将结果存储在 S3 上。

4.2.3 定价

按需付费，价格与所使用的吞吐率、数据存储空间、传输量等因素相关，详见[46]。

4.2.4 产品细节

1) 数据模型

DynamoDB 的数据模型由表、项和属性组成，表可包含多个项，每一项可有一个或多个属性。

- 属性（Attribute）：属性是名值对，属性名必须是字符串，但其值可以是字符串、数值、字符串集合或数值集合。例如：

```
"ImageID" = 1
"Title" = "flower"
```

```
"Tags" = "flower", "jasmine", "white"
"Ratings" = 3, 4, 2
```

- 项（Item）：项是属性的集合，以主键相区分，类似传统数据库中的记录概念。项的属性是一组名值对，顺序不固定。项的属性可以是稀疏和可选（除了主键属性）的，与同一表中的其它项的属性无关。与传统数据库不同的是，表没有 schema。项存储在表中，且必须指定至少一个属性作为主键。主键、项及表的关系如图 4-1 所示。

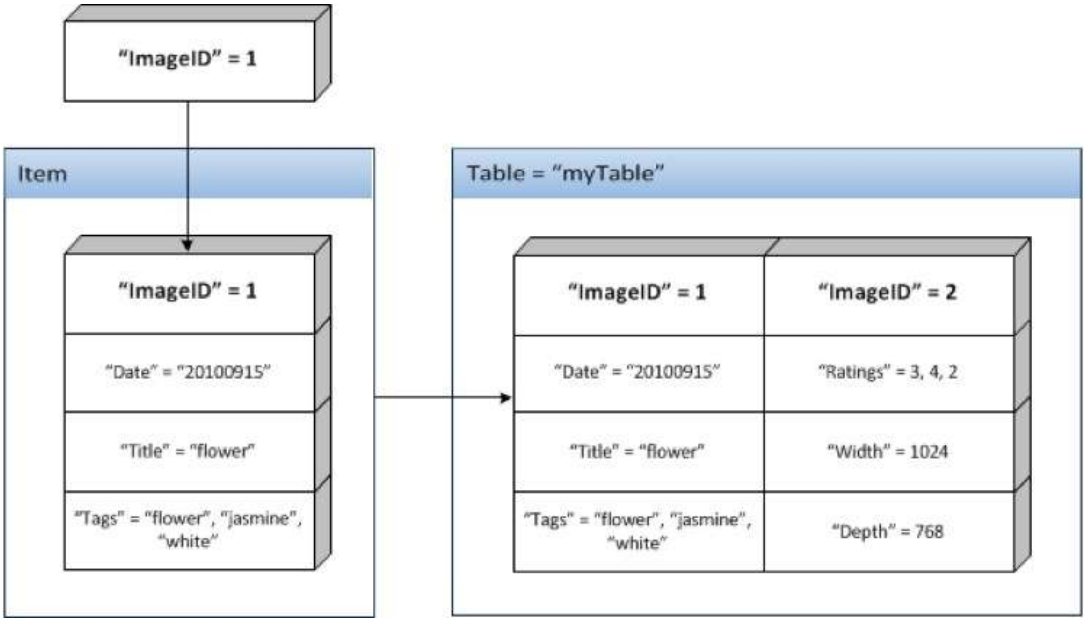

图 4-1　主键、项及表的关系

图中，ImageID 为主键，表名为“myTable”，项名为空，以主键 ImageID 定义。

- 表（Table）：同一表中的所有项具有相同的主键模式，项的主键值唯一。将数据写入 DynamoDB 的第一步就是创建表、指定表名与主键。表 4-2 是一个 DynamoDB 表的例子。

表 4-2　DynamoDB 表的例子

Table: My Images						
Primary Key	**Other Attributes**					
ImageID = 1	ImageLocation = https://s3.amazonaws.com/bucket/img_1.jpg	Date = 1260653179	Title = flower	Tags = Flower, Jasmine	Width = 1024	Depth = 768
ImageID = 2	ImageLocation = https://s3.amazonaws.com/bucket/img_2.jpg	Date = 1252617979	Rated = 3, 4, 2	Tags = Work, Seattle, Office	Width = 1024	Depth = 768

ImageID = 3	ImageLocation = https://s3.amazonaws.com/bucket/img_3.jpg	Date = 1285277179	Price = 10.25	Tags = Seattle, Grocery, Store	Author = you	Camera = phone
ImageID = 4	ImageLocation = https://s3.amazonaws.com/bucket/img_4.jpg	Date = 1282598779	Title = Hawaii	Author = Joe	Colors = orange, blue, yellow	Tags = beach, blanket, ball

2) 访问模型

DynamoDB 表只会为主键创建索引，可用来在多台服务器上哈希分区数据。主键可以是一个或两个属性的组合。对组合主键，一个属性作为哈希分区属性，另一个作为范围属性，例如，一个状态更新表的主键为“UserID”和“Time”，其中，UserID 为哈希属性，用来在多台服务器上分区数据以平均负载，Time 为范围属性。该例中，用户可以运行查询来获取：

- 以 UserID 和 Time 值标识的某个特定项；
- 某个 UserID 的所有项（哈希 bucket）；
- 某个 UserID 的某个特定时间范围内的所有项；只有在哈希分区属性已指定的情况下，才能使用范围属性。

DynamoDB 提供了若干 API，包括表的 Create、Update、Delete、Describe 操作，数据项的读写（Get & Put）、Update、Delete、查询（Query & Scan）及批量读写操作[48]。

4.3 Amazon SimpleDB

4.3.1 简介

Amazon SimpleDB[7]是具有高可用性及灵活性的非关系型数据库服务，开发人员只需通过 Web Service 存储并查询数据项，其它的事情都由 SimpleDB 管理。SimpleDB 自动将用户数据备份到多个物理分布的地点，以实现高可用性。用户可以随时更改数据模型，数据可以自动重建索引。

DynamoDB 和 SimpleDB 都是非关系型数据库服务，两者的区别主要是：

- DynamoDB 关注于提供无缝的扩展性和快速且可预知的性能，用户数据会被自动分布到多台服务器上以满足扩展的需求。DynamoDB 表中存储的数据没有数量限制，随着数据量的增长，用户可以扩展请求和存储容量来应对。
- SimpleDB 适合于对负载的扩展需求不高，但需要灵活性查询的情况。SimpleDB 自动为所有项属性建立索引，较之 DynamoDB 能提供更多的查询功能。SimpleDB 的表有 10 GB 的大小限制，如果需要超过此限制，可以手动分区数据，并存到额外的 SimpleDB 表中。

4.3.2 特点

SimpleDB 具有如下特点：

- 省心：用户只需关注业务应用本身，而 SimpleDB 会自动完成数据库管理工作，包括部署、软硬件维护、备份、创建索引及性能调优等。
- 高可用性：SimpleDB 自动为每个数据项创建多个地理上分布的备份，一旦某个副本出错，SimpleDB 会自动 failover 到另一份副本上。
- 灵活性：SimpleDB 的表可随时添加属性，并提供 consistent（强一致性，保证读取最新的数据）和 eventually consistent（尽量优化读取的低延迟和高吞吐率性能，高并发读写时，读取的数据可能不是最新，可短期内多次读取以得到最新数据，默认方式）两种读模式。
- 易用：SimpleDB 提供流线型的数据访问和查询功能，类似传统关系型数据库中的集群，通过 API 调用用户可以快速添加、获取并编辑数据。
- 与其它 AWS 集成：SimpleDB 可与 EC2 和 S3 结合使用，如开发人员可将应用运行在 EC2 上，把数据对象存储在 S3 上，并使用 SimpleDB 从 EC2 的应用中查询对象元数据，再从 S3 上获取实际数据对象。用户也可将 SimpleDB 和 RDS 结合使用，满足应用既需要关系型又需要非关系型数据库的需求。SimpleDB 和其它 AWS 在同一 Region 内部的数据传输是免费的。S3 采用的是密集型存储设备，一般可存放大文件或对象等原始数据，而 SimpleDB 的存储设备对访问速度要求更高，SimpleDB 中一般存放较小的数据元素或文件指针等，其会为数据创建索引以便快速查询数据。
- 安全：SimpleDB 提供 HTTPS 通信方式，并可与 AWS IAM 结合提供用户或组级别的访问权限控制。
- 廉价：按需付费。

4.3.3 定价

按需付费，具体价格与所选用的 Region、机器使用率、存储、数据传输量等有关[7]。

4.3.4 产品细节

1) 数据模型

SimpleDB 采用的数据模型，使得存储、管理和查询结构化数据变得容易。用户将数据存储在域（domain）中，并可对某一域中所有数据执行查询操作。域是项（item）的集合，项是若干属性-值对的集合。与传统关系型数据库相比，域类似表，项类似记录，属性类似列。如表 4-3 所示，域名为“customers”，每一行就是一个项，每一列头（如 CustomerID、City 等）表示一个属性，下面为其值。

表 4-3 SimpleDB 的一个例子

CustomerID	First name	Last name	Street address	City	State	Zip	Telephone

123	Bob	Smith	123 Main St	Springfield	MO	65801	222-333-4444
456	James	Johnson	456 Front St	Seattle	WA	98104	333-444-5555

若要将表 4-3 中的记录插入 SimpleDB，操作语句类似：
PUT (item, 123), (First name, Bob), (Last name, Smith), (Street address, 123 Main St.), (City, Springfield), (State, MO), (Zip, 65801), (Telephone, 222-333-4444) PUT (item, 456), (First name, James), (Last name, Johnson), (Street address, 456 Front St.), (City, Seattle), (State, WA), (Zip, 98104), (Telephone, 333-444-5555)

与传统数据库很不同的是，SimpleDB 允许用户在插入数据后，再为某些记录另添加新的属性，而无需删除原有数据、重新建表、重导入数据、重建索引等操作。

2) SimpleDB API

SimpleDB 提供了若干简单的 API，实现写、建索引和查询数据的功能，包括：

- *CreateDomain* — 创建域。
- *DeleteDomain* — 删除域。
- *ListDomains* — 列举所有域。
- *DomainMetadata* — 获取域的元数据，如创建时间、存储信息（项名及属性的数目）、数据集大小等。
- *PutAttributes* — 添加或更新项及其属性，或者为已有的项添加新的属性-值对。项会自动建索引。
- *BatchPutAttributes* — 可一个调用中最多执行 25 条 PutAttributes 操作。
- *DeleteAttributes* — 删除项、属性或属性值。
- *BatchDeleteAttributes* — 可一个调用中最多执行 25 条 DeleteAttributes 操作。
- *GetAttributes* — 获取项及所有或部分的属性与值。
- *Select* — 查询数据集，类似 select target from domain_name wherequery_expression 查询语句，支持=、!=、<、>、<=、>=、like、not like、between、is null、is not null 及 every()操作符。例如，select * from mydomain where every(keyword) = ‘Book’，对查询结果排序使用 SORT 操作符，对查询结果计数使用 Count 操作符。

4.4 Amazon ElastiCache

4.4.1 简介

Amazon ElastiCache Web Service[49]使得在 cloud 中部署、操作、扩展 in-memory 缓存变得容易。ElastiCache 提高了 Web 应用的性能，允许用户从快速、受管、in-memory 的缓存系统中获取信息，而非完全依赖于基于磁盘的数据库。ElastiCache 与 Memcached（一个广泛应用的内存对象缓存系统）协议兼容，因此，现有 Memcached 环境中的代码、应用和工具

可以无缝地用于 ElastiCache。

ElastiCache 简化了 in-memory 缓存的管理、监控和操作，用户只需关注应用逻辑本身，用户可数分钟内在自己的应用框架中添加 in-memory 缓存。通过 AWS 管理控制台，用户可启动一个由多个缓存节点（Cache Node）组成的缓存集群（Cache Cluster），每个节点上都运行 Memcached 软件。通过增删缓存节点，可以快速地向上或向下扩展缓存集群的内存数。ElastiCache 会自动检测并替换失效的缓存节点，避免因数据库过载而影响网站或应用的访问时间。通过与 CloudWatch 集成，ElastiCache 增强了对缓存节点关键性能参数的监控。

用户可使用 AWS 管理控制台或 ElastiCache API 来启动缓存集群，并指定缓存集群名、缓存节点类型及数目。用户可使用自己喜好的 Memcached 客户端[50]或编程语言连接缓存节点，用户 Memcached 代码与大部分客户端都可不经改动地用于 ElastiCache。

4.4.2 特点

Amazon ElastiCache 具有如下特点：

- 参数预配置：ElastiCache 的节点预配置了一组参数以用户满足所选择缓存节点类型，这样，就可以直接启动缓存集群了，用户也可以通过缓存参数组（Cache Parameter Group）来实现额外的控制。
- 自动故障检测与恢复：ElastiCache 监控缓存集群的健康情况，一旦发现网络分区、主机软硬件故障等，会自动替换缓存节点。替换的缓存节点和失效节点具有相同的 DNS 名。
- 详尽的监控与计量：ElastiCache 免费提供了针对缓存节点的详细 CloudWatch 计量参数，包括内存/计算资源使用率、缓存 hit、缓存 miss、缓存连接数等。
- 自动打补丁：ElastiCache 为缓存软件系统打补丁，用户可通过缓存引擎版本管理模块（Cache Engine Version Management[51]）来控制是否或何时为缓存集群打补丁。
- 易于扩展：通过 AWS 管理控制台或 API 调用可快速（分钟级别）增删缓存节点，以实现向上或向下扩展内存资源。
- 易于部署：通过 AWS 管理控制台或 API 调用，可快速（分钟级别）部署一个 Memcached 兼容的缓存环境，无需操心缓存软件架构的部署、安装与维护任务。
- 与其它 AWS 集成：ElastiCache 可与 RDS、SimpleDB、EC2、CloudWatch、SNS（Simple Notification Service）[52]等集成。例如，EC2 上的应用可安全低延迟地连接到同一 Region 中的 ElastiCache 集群，使用 CloudWatch 监控缓存使用情况，配置 Amazon SNS，以便在 ElastiCache 节点故障恢复后收到通知邮件。
- ElastiCache 提供了 Web Service 接口配置防火墙参数，来控制对缓存集群的网络访问。
- 成本经济：按需使用并收费（小时级别），可根据应用需求向上或向下无缝扩展缓存节点。

4.4.3 缓存节点类型

目前，Amazon ElastiCache 支持如下缓存节点类型：

- 标准类型，如表 4-4 所示。

表 4-4　标准类型缓存节点

类型	内存	ECU*	平台	I/O
小型（cache.m1.small）	1.3 GB	1[1]	64 位	中等
大型（cache.m1.large）	7.1 GB	4[2]	64 位	高
超大型（cache.m1.xlarge）	14.6 GB	8[3]	64 位	高

* 一个 ECU 提供了等同于 1.0-1.2 GHz 2007 Opteron 或 2007 Xeon 处理器的能力。

- 高内存类型，如表 4-5 所示。

表 4-5 高内存类型缓存节点

类型	内存	ECU	平台	I/O
超大型（cache.m2.xlarge）	16.7 GB	6.5[4]	64 位	高
双倍超大型（cache.m2.2xlarge）	33.8 GB	13[5]	64 位	高
四倍超大型（cache.m2.4xlarge）	68 GB	26[6]	64 位	高

- 高 CPU 类型，如表 4-6 所示。

表 4-6 高 CPU 类型缓存节点

类型	内存	ECU	平台	I/O
超大型（cache.c1.xlarge）	6.6 GB	26[7]	64 位	高

4.4.4 定价

按需使用并收费，具体价格与所在的 Region、缓存节点类型、传输的数据量有关[49]。

4.4.5 产品细节

1) 主要应用场景

Amazon ElastiCache 可显著提高 read-heavy 应用（如社交网络（Social Networking）、游戏、媒体分享、Q&A portal 等）或计算密集型应用（如推荐引擎等）的延迟和吞吐率性能。ElastiCache 通过将关键数据存储在内存中来降低访问延迟，提高应用性能，缓存信息包括 I/O 密集型数据库查询结果或计算密集型计算结果等。

2) 操控 ElastiCache

用户可通过 AWS 管理控制台或 Web Service API 来创建、删除、修改 ElastiCache 缓存集群。一个 ElastiCache 缓存集群是一组缓存节点的逻辑集合，每个节点上均运行着

[1] 1 个虚拟核上运行 1 个 EC2 计算单元（ECU）
[2] 2 个虚拟核上各运行 2 个 ECU
[3] 4 个虚拟核上各运行 2 个 ECU
[4] 2 个虚拟核上运行 3.25 个 ECU
[5] 4 个虚拟核上各运行 3.25 个 ECU
[6] 8 个虚拟核上各运行 3.25 个 ECU
[7] 8 个虚拟核上运行 2.5 个 ECU

Memcached 软件。用户可使用缓存安全组（Cache Security Group）[53]来控制缓存集群的访问和安全设置，并使用缓存参数组（Cache Parameter Group）[54]来调优部署。

3) 移植现有应用

对 Memcached 上的现有应用，可依照如下步骤轻松移植到 ElastiCache 上：

a) 创建某一缓存节点类型的缓存集群，指定缓存节点的数目；
b) 配置缓存安全组，允许运行应用的 EC2 实例（或应用服务器）访问缓存集群；
c) 获取步骤 a)中创建的缓存几点 endpoint（或 DNS 名）信息，具体方式为：访问 AWS 管理控制台的 Amazon ElastiCache tab 页，在缓存集群中的“Nodes”部分，点击“Copy Node Endpoints”按钮，也可使用 ElastiCache 的 DescribeCacheClusters API 获取；
d) 配置 EC2 实例访问缓存集群：更新 Memcached 客户端的库中的 Memcached 配置文件，加入步骤 c)中获取的 endpoint 信息。

以上步骤结束后，建议进行完全的测试以完成移植工作。

4) 在不同的 AZ 中设置冗余的缓存集群

Amazon ElastiCache 会主动监控缓存节点的健康状况，并在其故障时予以替换。但因为缓存的特性，新替换的缓存节点是空的（即所谓“cold”），然后根据负载的情况，稍后重新装载数据（即所谓“warming up”）。ElastiCache 的自动替换功能受限在单一 AZ 内，如果应用对故障恢复或缓存节点 warm up 时间比较敏感，或用户想在获得 AZ 级别的容错能力，则需要在不同的 AZ 上部署冗余的 ElastiCache 集群。

用户可配置应用向跨 AZ（主从 AZ）的多个缓存节点写缓存，一旦主 AZ 中的某个缓存节点失效，应用可以在 ElastiCache 恢复该节点的同时，由从 AZ 的相应缓存节点直接读取数据。

第五章 部署与管理类（Deployment & Management）

5.1 AWS Identity and Access Management (IAM)

5.1.1 简介

AWS 身份与访问管理服务（Identity and Access Management，IAM）[55]提供了对 AWS 服务和资源的安全访问控制，使用 IAM 可在 AWS 中创建并管理用户、角色、权限与安全凭证，即使不在 AWS 之内，IAM 也可赋予受管用户对 AWS 资源的访问权限。因为 IAM 和其它 AWS 服务实现了无缝集成，所以如果使用 AWS，IAM 可以提供更多的安全、灵活性和控制能力。

IAM 可在用户公司目录（corporate directory）和 AWS 服务之间建立 identity federation，这样，就可使用客户公司现有的用户身份信息对 AWS 资源授予安全直接的访问，而无需为这些用户创建新的 AWS 身份[56]。

AWS 管理控制台提供了 IAM 的操作入口。

5.1.2 功能

使用 IAM，可以：

- 管理 IAM 用户及其访问权限
  在 IAM 中创建用户并加入 AWS 账户中，为用户指定个人安全凭证（例如，access key、密码、Multi-Factor 认证设备[57]等）或请求临时安全凭证来访问 AWS 服务和资源。通过添加特定的条件可以控制用户如何使用 AWS，比如使用多少天、源 IP 地址、是否使用 SSL 或者是否通过 Multi-Factor 认证设备进行认证等。为使移动与基于浏览器的应用能安全访问 AWS 资源，可请求临时安全凭证，并仅赋予指定时间内访问特定 AWS 资源的权限。
  对 AWS 账户下的 IAM 用户可分组管理，每个组具有不同的权限。
- 管理 IAM 角色及其权限
  在 IAM 中可以创建角色（用户或 AWS 服务）并加入 AWS 账户中，通过权限管理可以控制角色能够执行哪些操作。
- 管理 federated 用户的访问
  IAM 具有精细访问控制的能力，能通过 identity federation 机制可以使客户企业内的现有身份信息（如用户）访问 AWS 管理控制台、AWS API 及资源，而无需为每个身份信息创建新的 IAM 用户。
  采用 identity federation，需先请求临时安全凭证，用于 AWS 访问的签名。临时安全凭证由短期 access key 及与之相关的 session token 组成。企业内部用户可按以前的用法来使用 access key，并在调用 AWS API 时附带上 token。临时安全凭证关联的权限最多与签发这些凭证的 IAM 用户一致，可通过显式的权限指派来予以限制，

IAM 用户能签发的临时安全凭证数目没有限制。

下面来看一个例子，某企业希望在所有员工的笔记本上运行一个应用，每天都会将员工指定的某个目录备份到 S3 上。此时，该企业可以运行一个小应用作为身份代理，为每个登录进入公司网络的用户申请一个临时安全凭证。该凭证授予了固定的权限（即对特定 S3 bucket/folder 的写权限）及权限有效期（如 12 小时）。这个凭证会被回传给员工笔记本上的备份应用，提供对 Amazon S3 安全直接的访问。

5.1.3 特点

IAM 具有如下特点与功能：

- 增强的安全性：IAM 能为每个用户授予唯一的安全凭证，指定可以访问哪些 AWS 服务 API 和资源。用户无法访问 AWS 资源，除非 IAM 显式地授予其权限。
- 精细控制：IAM 可以精细控制用户对特定 AWS 服务和资源的访问（如停止某个 EC2 实例或删除某个 S3 bucket）。
- 可靠性：与其它 AWS 一样，IAM 运行在 Amazon 全球网络基础架构与数据中心之上，具有高可靠性。

5.1.3 定价

AWS IAM 作为 AWS 账户的特性，是免费提供的服务，用户仅需为所使用的其它 AWS 服务付费。

5.1.4 产品细节

用户可通过 AWS 管理控制台、API[58]或命令行工具来使用 IAM，其中一些常用的 IAM API 描述如下：

- 管理用户
  - *CreateUser*：创建 IAM 用户，用户名为字符串，且在所属的 AWS 帐户中唯一；
  - *CreateGroup*：创建组；
  - *AddUserToGroup*：将 IAM 用户添加到指定的组中；
  - *RemoveUserFromGroup*：从指定组中删除 IAM 用户；
- 管理角色
  - *CreateRole*：创建 IAM 角色，角色名为字符串，且在所属的 AWS 账户中唯一；
- 管理权限
  - *PutUserPolicy*：添加（或更新）指定 IAM 用户的策略文档，策略文档包含一组权限；
  - *PutRolePolicy*：添加（或更新）指定 IAM 角色的策略文档。
  - *PutGroupPolicy*：添加（或更新）指定组的策略文档。
- 管理凭证
  - *CreateAccessKey*：创建指定 IAM 用户的一个唯一 AWS Access Key ID 及相应的 Secret Access Key。Access Key 允许 IAM 用户通过 API 直接调用 AWS 服务。
  - *GetSessionToken*：创建一个唯一的 AWS Access Key ID 及相应的 Secret Access

Key 和 token，使得 IAM 用户可在指定时间内调用本 API。

- *GetFederationToken*：为 federated 用户或应用创建一个唯一的 AWS Access Key ID 及相应的 Secret Access Key 和 token，并为之指定时间段和权限。

5.2 Amazon CloudWatch

5.2.1 简介

Amazon CloudWatch[10]监控 AWS 云资源（如 EC2 和 RDS DB 实例）及 AWS 上运行的应用，开发人员和系统管理员可使用 CloudWatch 来收集、跟踪计量数据（包括用户自定义的），对资源使用率、应用性能或操作情况有全面的了解，从而保证应用与业务的平滑运行。

用户可通过编程方式获取监控数据、查看图表、设置警报来帮助定位错误、显示趋势，并根据云环境的具体情况自动采取相应的操作。

5.2.2 功能

CloudWatch 可以实时监控 AWS 资源，包括 EC2 实例、EBS 卷、ELB 及 RDS DB 实例，计量数据包括 CPU 使用率、延迟、请求数等，用户也可添加自己的计量参数，如内存使用率、事务处理量、错误率等。

- 自动监控 AWS 资源，无需安装额外的软件
  - 对 EC2 实例的基本监控：预置了 10 个计量参数，5 分钟频度，免费；
  - 对 EC2 实例的详细监控：预置了 7 个计量参数，1 分钟频度，收费；
  - EBS 卷：预置了 8 个计量参数，5 分钟频度，免费；
  - ELB：预置了 10 个计量参数，1 分钟频度，免费；
  - RDS DB 实例：预置了 13 个计量参数，1 分钟频度，免费；
  - SQS 队列：预置了 8 个计量参数，5 分钟频度，免费；
  - SNS：预置了 4 个计量参数，5 分钟频度，免费；
  - ElastiCache 节点：预置了 29 个计量参数，1 分钟频度，免费；
  - DynamoDB 表：预置了 7 个计量参数，5 分钟频度，免费；
  - Storage Gateway[59]：预置了 11 个网关计量参数及 5 个存储卷计量参数，5 分钟频度，免费；
  - Elastic MapReduce 工作流：预置了 23 个计量参数，5 分钟频度，免费；
  - Auto Scaling 组：预置了 7 个计量参数，1 分钟频度，可选、收费；
  - 还可启动计量参数来监控 AWS 收费情况，计量参数的数目依 AWS 服务不同而不同，免费[60]。
- 通过 Put API 请求提交自定义的计量参数。
- 针对计量参数设置警报，当计量参数值超过指定阈值时，发送通知或采取自动化的对应操作。
- 每个计量参数都提供了图表和统计数据分析，用户可从 CloudWatch 的 dashboard 监控 AWS 资源，并查看所有的告警。
- Auto Scaling 依赖于 CloudWatch 的计量参数来自动增删 EC2 实例。

5.2.3 定价

除了第 5.2.2 节中描述的 AWS 资源的免费计量参数外，其它计量参数的收费标注为：

- 每个设置的计量参数每月 0.50 美元；
- 每个设置的告警每月 0.10 美元。

5.2.4 常用场景

一旦注册了 Amazon EC2，就自动注册了 CloudWatch。

- 使用 CloudWatch 监控 EC2 实例
  CloudWatch 基本监控模板以 5 分钟的频度从每个 EC2 实例收集并报告 CPU 使用率、数据传输量、磁盘使用情况等计量信息。CloudWatch 详细监控模板提供同样的功能，但频度为 1 分钟，并按 EC2 AMI ID 和实例类型来汇集数据。如果用户还使用了 Auto Scaling 或 ELB 服务，CloudWatch 还可以按 Auto Scaling 组或弹性负载均衡器来汇集 EC2 实例的计量数据。监控数据保留两星期，即使用户已经停止使用 AWS 资源了。CloudWatch 的使用步骤为：
  a) 登录 AWS 管理控制台；
  b) 在 Amazon EC2 tab 页，点击 Launch Instances 按钮；
  c) 选择一个 AMI 来运行一个实例，选择 Key Pair（密钥对）并配置防火墙；
  d) 在最后一步，点击“Enable CloudWatch Detailed Monitoring for this instance”选择框；
  e) 点击 Launch 按钮；
  f) 几分钟内，刚启动的实例即可显示开始运行了；
  g) 对正在运行的实例也可配置 CloudWatch 详细监控，只需在 Amazon EC2 tab 页右键单击该实例，并选择“Enable Detailed Monitoring”即可。
- 使用 CloudWatch 监控其它 AWS 资源
  CloudWatch 还可自动监控如下计量参数[61]：
  - ELB：请求数、延迟；
  - EBS 卷：读写延迟；
  - RDS DB 实例：可用内存、可用存储空间；
  - SQS 队列：发送与接收的消息数；
  - SNS：发布与传输的消息数。
- 使用 CloudWatch 监控自定义的计量参数
  使用 Put API 调用，用户可以快速发送并存储任何数量的自定义计量参数，自定义计量参数的数据、图表、告警等以 1 分钟频度获取[61]。
- 访问 CloudWatch 计量参数
  - 在 AWS 管理控制台的 CloudWatch tab 页，点击 Metrics 链接，将列出所有 AWS 资源及自定义计量参数，选择一个计量参数；
  - 将显示该计量参数的交互式图表，选择一个时间段，以平均、最小、最大、求和或取样数来显示数据值；
  - 选择另一个计量参数加到此图表上，来发现关联行为模式；
  - 点击 Create Alarm 按钮，当计量参数达到某个设定值时，可触发自动通知或 Auto Scaling 动作；

- 点击 Dashboard 链接，查看所有告警和 AWS 资源计量参数的总体状态。

5.3 Amazon Elastic Beanstalk

5.3.1 简介

AWS Elastic Beanstalk[62]使用户可以方便快速地在 AWS 云上部署并管理应用。用户上传应用后，Elastic Beanstalk 会自动处理部署细节，如资源分配、负载均衡、自动扩展及应用健康监控，同时，用户对应用所使用的 AWS 资源仍具有完全的控制权。

Elastic Beanstalk 默认的应用运行环境配置为一个 Amazon EC2 微型实例（应用服务器）及一个 Amazon ELB，并在应用峰值时自动扩展 EC2 实例。每个 EC2 实例都来自与 Amazon 机器镜像文件（Amazon Machine Image，AMI），AMI 包含了创建新的服务器实例所需的全部信息。Elastic Beanstalk 默认使用 Amazon Linux AMI 或 Windows Server 2008 R2 AMI，这些 AMI 包含了 Web 服务器及应用服务器所需的所有软件（如 Linux、Apache、PHP 等）。

5.3.2 功能

Elastic Beanstalk 为运行的应用提供了如下管理功能：

- 在运行环境中部署新版本的应用或回滚到前一版本；
- 访问内置的 CloudWatch 监控计量参数，如 CPU 平均使用率、请求数、平均延迟等；
- 当应用的健康状况变化或增减应用服务器时，通过 Amazon SNS 来接收 e-mail 通知；
- 无需登录应用服务器即可访问服务器的日志文件；
- 通过一条命令即可快速重启所有 EC2 实例上的应用服务器。

借助 Elastic Beanstalk，开发者可以完全控制其应用所使用的 AWS 资源，从 Elastic Beanstalk 管理控制台修改默认的配置参数，还可完成各种功能，包括：

- 选择最适合自己应用的 EC2 实例类型，以满足对 CPU 和内存的需求；
- 多种数据库和存储选项，如 Amazon RDS、DynamoDB、SimpleDB、Microsoft SQL Server、Oracle、IBM DB2 或 Informix；
- 可直接登录 EC2 实例，快速进行故障诊断；
- 可将应用快速部署在多个 AZ 上，以提高可靠性；
- 在负载均衡器上采用 HTTPS 协议，以加强应用的安全性；
- 调整应用服务器的设置（如 JVM 参数），并传递环境变量；
- 在 EC2 上并行运行其它应用模块，如内存缓存服务等；
- 调整 Auto Scaling 设置来控制计量参数和阈值，以决定何时增删 EC2 实例；

不同语言的开发人员可使用 Elastic Beanstalk 来部署各自的应用。

- 对.NET 开发人员

  使用 Elastic Beanstalk 来部署.NET 应用的步骤如下：

  a) 使用 Microsoft Visual Studio 等工具创建.NET 应用；

  b) 使用 Visual Studio 的 AWS 工具包[63]将应用发布到 AWS Elastic Beanstalk；或者，

将代码打包成 Microsoft Web Deploy .zip 包，并使用 AWS 管理控制台上传到 AWS Elastic Beanstalk；

c) Elastic Beanstalk 将在后台部署一个负载均衡器，并将 Microsoft Web Deploy 包部署到一个或多个运行 Windows Server 2008 R2 及 IIS 7.5 的 EC2 实例上；

d) 数分钟后，用户就可以通过特定的 URL（如 http://myapp.elasticbeanstalk.com/）来访问应用了。

- 对 PHP 开发人员

  使用 Elastic Beanstalk 来部署 PHP 应用的步骤如下：

  a) 创建 PHP 应用；

  b) 使用 AWS 管理控制台、命令行接口或 Web Service API 来创建 Elastic Beanstalk PHP 环境；Elastic Beanstalk 将在后台部署一个负载均衡器和 EC2 资源；

  c) 安装并配置 Git；

  d) 使用 Git 来提交代码更改[64]；Elastic Beanstalk 将应用部署到一个或多个运行 Apache HTTP 服务器的 EC2 实例上；

  e) 数分钟后，用户就可以通过特定的 URL（http://myapp.elasticbeanstalk.com/）来访问应用了。

- 对 Java 开发人员

  使用 Elastic Beanstalk 来部署 Java 应用的步骤如下：

  a) 使用 Eclipse 等创建 Java 应用；

  b) 将代码打包成标准的 Java Web Application Archive 包（WAR 文件）；

  c) 使用 AWS 管理控制台、Eclipse 的 AWS 工具包[65]、Web Service API 或命令行接口将 WAR 文件上传到 Elastic Beanstalk；

  d) Elastic Beanstalk 将在后台部署一个负载均衡器，并把 WAR 文件部署到一个或多个运行 Apache Tomcat 应用服务器的 EC2 实例上；

  e) 数分钟后，用户就可以通过特定的 URL（http://myapp.elasticbeanstalk.com/）来访问应用了。

5.3.3 特点

Elastic Beanstalk 具有如下特点：

- 易于上手：用户可通过 AWS 管理控制台、Git 部署或 IDE（如 Eclipse 或 Visual Studio）来上传应用，Elastic Beanstalk 会自动处理部署的细节，包括资源分配、负载均衡、自动扩展及健康监控。
- 自动扩展：Elastic Beanstalk 基于 Auto Scaling 的默认设置自动向上或向下扩展应用，用户可根据应用的特定需求而调整 Auto Scaling 设置，如使用 CPU 使用率来触发 Auto Scaling 动作，这样，应用可以轻松应对负载或流量峰值情形。
- 完全控制：用户通过 Elastic Beanstalk 可完全控制自己应用所使用的 AWS 资源，如浏览日志文件、监控应用健康情况、调整自动扩展规则、设置 email 通知，甚至可通过 Elastic Beanstalk 控制台传入环境变量。
- 灵活性：用户可自由选择 EC2 实例的类型及数据库等。
- 可靠性：Elastic Beanstalk 运行在 Amazon 可信网络框架及数据中心中，保证了应用的高可靠性和可用性。

5.3.4 定价

Elastic Beanstalk 是免费，用户只需为应用所使用的底层 AWS 资源付费，如 EC2、S3、带宽、数据库、ELB 等[62]。

5.4 AWS CloudFormation

5.4.1 简介

AWS CloudFormation[66]使开发者和系统管理员能按顺序且以可预测的方式来创建、管理、部署并更新一组相关的 AWS 资源。

用户可以使用 AWS CloudFormation 的模板[67]或自定义模板来描述运行应用所需的 AWS 资源、相关的依赖或运行参数。CloudFormation 负责处理 AWS 服务的部署顺序及依赖关系，用户无需关心。部署完成后，用户对 AWS 资源的修改和更新具有可控性，对 AWS 基础架构的版本控制和普通软件的版本控制一致无二。

用户可通过 AWS 管理控制台、CloudFormation 命令行工具或 API 来部署并更新模板及其相关的资源栈。CloudFormation 免费使用，用户只需为所使用的其它 AWS 资源付费。

5.4.2 功能

AWS CloudFormation 的使用步骤如下：

1) 注册 AWS CloudFormation，即可自动注册 AWS CloudFormation 支持的所有 AWS 服务，用户仅为所使用的服务付费。注册后，用户可使用 AWS 管理控制台、命令行工具或 API 来使用 AWS CloudFormation。
2) 创建 CloudFormation 栈，来部署应用所需的所有资源，步骤为：
   a) 在 AWS 管理控制台的 CloudFormation tab 页选择要使用的模板。CloudFormation 提供了常用开源软件应用的样例模板，采用样例模板集成了多种 AWS 资源，提供了地理冗余、横向扩展及告警的最佳实践。用户也可加载存储在 S3 或本地磁盘上的自定义模板。
   b) 用户可使用模板提供的默认参数值，也可通过重载模板参数来定制 AWS 资源栈，如数据库或应用名称、密码、端口、EC2 实例类型等。完成后，CloudFormation 负责 AWS 资源栈的创建与部署。
3) 用户可从 AWS 管理控制台查看 AWS 资源栈中包含的所有资源。
4) 如果需要，可基于已创建的模板或样例模板生成更多资源栈。模板为文本文件，可在 CloudFormation 之外创建并管理，因此，可通过 email、源码控制库或 S3 等服务分享模板。
5) 当软件更新或配置改变后，用户可随时更改运行中的资源栈，步骤为：
   a) 根据资源栈的配置改变情况而修改模板，此时可使用软件版本控制等方式来管理更改。
   b) 通过 AWS 管理控制台、命令行工具或 API 将更新的模板提交到 AWS

CloudFormation，CloudFormation 负责使更改生效等后续操作。

5.4.3 特点

AWS CloudFormation 具有如下特点：

- 支持大量 AWS 资源

  AWS CloudFormation 支持许多 AWS 资源，来构建高可用性、可靠性和可扩展的 AWS 体系架构，以满足应用的需要。CloudFormation 支持的 AWS 资源有：
  - EC2 实例：按需实例、Spot 实例和保留实例
  - Elastic Block Store（EBS）卷
  - ELB
  - 弹性 IP 地址
  - EC2 安全组
  - Auto Scaling 组
  - RDS 实例
  - RDS 安全组
  - ElastiCache 集群
  - ElastiCache 安全组
  - ElastiCache 参数组
  - Elastic Beanstalk
  - CloudWatch 告警
  - S3 bucket
  - SimpleDB 域
  - SQS 队列
  - SNS 话题
  - SNS 订阅
  - Route 53 DNS 记录
  - CloudFront 分发包与多源支持
  - IAM 用户与组
  - IAM 策略
  - VPC、子网、网关、路由表与网络 ACL
  - DynamoDB 表
- 易于使用

  CloudFormation 负责部署应用所需的 AWS 资源，用户可定义运行时所需的依赖关系或参数，而无需关心 AWS 资源的部署顺序及如何处理依赖关系，同时，CloudFormation 还提供了大量样例模板[67]供用户选择，包括：
  - WordPress（博客）
  - Tracks（项目跟踪）
  - Gollum（GitHub 使用的 Wiki）
  - Drupal（内容管理）
  - Joomla（内容管理）
  - Insoshi（社交应用）
  - Redmine（项目管理）
  - 使用单个 AWS 资源及特性的样例模板

- 透明与开放：模板是 JSON 格式的文本文件，可通过普通的源码控制软件进行管理，存储在私有或公开的位置，如 S3 或 email。对资源栈中的任意 AWS 资源，用户都有完全的控制和修改权限。
- 参数定制化：用户可在运行时定制模板参数，如 RDS 数据库大小、EC2 实例类型、数据库与 Web 服务器端口号等。用户还可根据应用所部署的 Region 不同，而设置不同的模板参数，以生成不同的资源栈，如部署在 US Region 较之 Europe Region 需要更大的带宽等，这样，各个不同 Region 可以为某个应用设置不同的参数，但整个应用在所有 Region 上仍然保持部署的一致性。
- 进度跟踪：CloudFormation 通过 AWS SNS 来发布进度情况，用户可使用 e-mail 等方式跟踪资源栈的创建与删除进度。

5.4.4 定价

使用 AWS CloudFormation 不会产生额外的费用，用户只需为使用 CloudFormation 创建的 AWS 资源（如 EC2 实例、ELB 等）付费。

5.4.5 产品细节

1) 创建 AWS CloudFormation 模板

CloudFormation 模板是 JSON（Javascript Object Notation）格式的文本文件，描述了执行一个应用或服务及其内部互联所需的 AWS 体系框架支持。模板记载资源之间的关系，如 EC2 必须配合 ELB 使用，或 EBS 卷必须和绑定的 EC2 在同一 AZ 中。CloudFormation 模板可参数化，以满足同一模板用于不同配置要求的部署场景。模板亦提供输出参数，以便于配置交互，例如，在新的应用实例化后，模板可以返回 ELB endpoint 的 URL，用户可用来连接新实例化的应用。模板中的所有 AWS 资源以逻辑名相区别，因此使用同一模板创建的多个资源栈不会存在 AWS 资源命名冲突问题。

模板的 JSON 结构定义如代码清单 5-1 所示：

代码清单 5-1 CloudFormation 模板的 JSON 结构定义

```
{
   "Description" : "模板用途的文本描述",
   "Parameters": {
      // 定制模板的输入参数，因部署而异
   },
   "Resources" : {
      // AWS 资源及其之间关系
   },
   "Outputs" : {
      // 栈创建者可见的输出值
   },
   "AWSTemplateFormatVersion" : "2010-09-09"
}
```

创建一个 EC2 实例的简单模板如代码清单 5-2 所示。

代码清单 5-2　创建一个 EC2 实例的模板例子

```
{
   "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.”
   "Parameters" : {
      "KeyPair" : {
         "Description" : "The EC2 Key Pair to allow SSH access to the instance",
         "Type" : "String"
      }
   },
   "Resources" : {
      "Ec2Instance" : {
         "Type" : "AWS::EC2::Instance",
         "Properties" : {
            "KeyName" : { "Ref" : "KeyPair" },
            "ImageId" : "ami-75g0061f"
         }
      }
   },
   "Outputs" : {
      "InstanceId" : {
          "Description" : "The InstanceId of the newly created EC2 instance",
          "Value" : { "Ref" : "Ec2Instance" }
       }
   },
   "AWSTemplateFormatVersion" : "2010-09-09"
}
```

2)　使用 AWS CloudFormation 模板来创建并管理 AWS 资源栈

实例化模板而得到的资源集合称为栈，栈是由 AWS CloudFormation 服务基于模板及相应参数而创建的。模板规定了以怎样的顺序来创建哪些 AWS 资源，创建顺序由模板中声明的资源的依赖关系决定。这些依赖关系有些是隐式的，例如要将 EBS 卷和 EC2 实例连接起来，就需要先创建 EBS 卷。另一些依赖关系是显式的，例如，如果使用 Auto Scaling 组部署的应用需要访问 RDS 实例，则 RDS 实例必须在 EC2 实例之前创建，这时就需要在模板中定义此依赖关系。

在栈创建时，CloudFormation 负责记录资源在模板中的逻辑名称（如“myServer”）与实例化后的实际名称（如 EC2 实例名“i-19d3ac161”）之间的映射关系，可通过 API 获取栈的创建状态资及资源名称的映射关系。此外，CloudFormation 会为 EC2 资源（如 EC2 实例和 EBS 卷）打上栈名的标签。

栈的更新是通过提供一个包含所有资源更新配置的模板来实现的。例如，可以更改 CloudWatch 的告警阈值或更新实例的 AMI。CloudFormation 负责栈内资源的更新，大部分情况下，更新都不会影响运行中的应用，但如果更新无法动态实现（如更新 EC2 实例的

AMI），CloudFormation 就会创建一个新的实例加入栈中，并在更新成功后删除旧的资源。资源更新支持回滚操作。在创建栈时，出于调试的目的，可暂时关闭回滚操作。

AWS CloudFormation 的创建、更新与删除操作可通过 AWS 管理控制台或 API[68]实现，若干常用 API 及其功能描述如下：

- *CreateStack*：创建一个新的栈，输入参数包括栈名及模板文件名（或 S3 URL）。创建成功后，栈将处于 CREATE_COMPLETE 状态。CloudFormation 会删除之前创建的资源，除非用户设定标志符，声明保留这些资源以用于调试用途。
- *ListStacks*：列出用户账号下的所有栈。可查看所有栈及其当前状态，如是否有正在执行的更新或创建操作、栈是否可用。
- *ListStackResources*：列出栈的所有 AWS 资源名及标识符。可用来查看栈创建的资源，基于 CloudFormation 的应用也可使用该接口来获取所部署的环境信息。
- *DescribeStackEvents*：列出 CloudFormation 为栈而生成的所有操作和事件，以了解栈的创建和删除进度。
- *UpdateStack*：更新一个已存在的栈，输入参数包括栈名及更新模板的文件名（或 S3 URL）。更新完成后，栈将处于 UPDATE_COMPLETE 状态。如果更新失败，CloudFormation 会自动回滚到原有模板描述的状态。

CloudFormation 与 Amazon SNS 实现了集成，在栈的创建、更新和删除过程中，用户可以收到状态通知，这样，其它程序可以获取 CloudFormation 的内部事件并予以响应，以参与栈的配置过程。

3) 使用 AWS 资源

CloudFormation 模板定义资源的例子如代码清单 5-3 所示。

代码清单 5-3 CloudFormation 模板定义资源的例子

```
"myVolume" : {
   "Type" : "AWS::EC2::Volume",
   "Properties" : {
      "Size" : "10",
      "SnapshotId" : "snap-7b8fd361",
      "AvailabilityZone" : "us-east-1a"
   }
}
```

代码清单 5-3 中，模板定义了一个 Amazon EBS 卷。卷的逻辑名为“myVolume”，其类型为“AWS::EC2::Volume”。

4) CloudFormation 脚本

AWS CloudFormation 提供了一组可部署到 EC2 实例中的脚本[70]，这些脚本提供了多种实用的功能，如读取栈内资源的元数据并用来配置应用、将包和文件部署到模板中列出的实例上、执行栈更新操作等。可用的 CloudFormation 脚本如下：

- cfn-get-metadata：获取模板中定义资源的元数据；
- cfn-init：下载并安装模板中描述的包和文件；
- cfn-signal：向栈的创建工作流发送信号，应用已经启动并等待就绪。

- cfn-hup：一个 daemon，监听栈的更新并执行应用自身的 hook 来响应这些更新。

CloudFormation 脚本可以独立使用，也可和 CloudInit 一起使用。CloudInit 是 Amazon Linux AMI[69]及其它一些 Linux AMI 所具有的一个特性，可在实例启动时使用 EC2 的 user-data 字段将配置操作传递给实例，使 EC2 实例的远程配置成为可能。

第六章 应用服务类（Application Services）

6.1 Amazon CloudSearch

6.1.1 简介

Amazon CloudSearch 是一个基于云的快速且高扩展性的搜索服务[71]，可被用户集成到自己的应用中。CloudSearch 具有高吞吐率和低延迟的特点，支持自由文本搜索、分面搜索（Faceted Search）、可定制的相关度分级、可配置的搜索字段、文本处理选项及近似实时的索引构建。开发者可使用 AWS 管理控制台创建搜索域并上传搜索源数据，CloudSearch 即可部署所需的资源并建立优化过的搜索索引。

随搜索源数据增长或查询率的变化，CloudSearch 会自动无缝地进行扩展，开发者也可更改搜索参数、精细调优搜索相关度，并随时应用新的设置而无需重新上传搜索源数据。

采用 CloudSearch 后，用户不再担心硬件部署、数据分区、软件补丁等管理型事务与弹性扩展的要求。Amazon CloudSearch 的使用步骤为：

1) 创建一个搜索域；
2) 配置搜索字段；
3) 上传用于搜索的源数据，CloudSearch 会自动为之建立索引；
4) 从用户的网站或应用提交搜索请求。

6.1.2 特点

Amazon CloudSearch 具有如下特点：

- 易于配置：用户可使用 AWS 管理控制台、命令行工具或 API 来上传用于搜索的数据，CloudSearch 自动生成索引字段列表及建议的配置。用户能方便地增删索引字段及定制搜索选项，如分面（Facet，用来细化筛选搜索结果的索引字段，如衣服的颜色、电视的尺寸等）。更改配置无需重传搜索源数据。
- 自动扩展：随着搜索数据量和查询量的变化，CloudSearch 会自动无缝地向上或向下扩展。CloudSearch 处理操作的 footprint 并部署搜索实例。
- 低延迟、高吞吐率：CloudSearch 将索引存储在内存中，并和 Amazon.com 的搜索引擎采用同样的 A9 技术。
- 易于管理：CloudSearch 负责软硬件的部署、安装与配置、软件打补丁及数据分区等管理操作，无需用户操心。
- 丰富的搜索特性：CloudSearch 可为结构化文本和纯文本建立索引并搜索，其包含用户期望的大部分搜索引擎功能，如分面搜索、自由文本搜索、布尔搜索、可定制的相关度分级及基于任意字段的结果排序等，CloudSearch 还可为更新数据近似实时地建立索引。
- 安全：CloudSearch 使用强加密方法进行用户认证，以防止对搜索域未经授权的访

问控制。CloudSearch 支持 HTTPS 协议，提供了 Web Service 接口来设置防火墙参数以控制对搜索域的网络访问。

6.1.3 定价

CloudSearch 每月按用户对以下几项的使用情况来收费[71]：

- 搜索实例类型
- 文档批量上传量
- 索引文档请求数
- 数据传输量

6.1.4 产品细节

本节将介绍搜索实例类型及 CloudSearch 的子服务，有关 CloudSearch 的更多细节请参考 CloudSearch 开发者指南[72]。

1) 搜索实例类型

搜索域为搜索源数据的集合，每个搜索域有一个或多个搜索实例，每个实例包含有限数量的 CPU 和内存资源用于数据索引构建和请求处理。搜索域的索引实例数目取决于域中的文档数量，以及搜索请求的数量和复杂度。

CloudSearch 决定为保证低延迟高吞吐率的搜索性能而必需的搜索实例大小与个数。当用户上传源数据时，CloudSearch 会构建一个索引，并选择适当的初始搜索实例类型以保证索引能存储在内存中。

随着数据量增长，CloudSearch 将扩展搜索域到一个更大的搜索实例类型上（如果已处于最大的搜索实例类型，则可以跨多个实例将索引分区存储）。相反的，当数据量缩减时，CloudSearch 可以将搜索域向下扩展到更少的搜索实例上（如果索引未分区，则选择更小的搜索实例类型）。

当搜索实例的 CPU 使用率超过 80%时，CloudSearch 会为搜索域自动添加一个搜索实例；当搜索实例的 CPU 使用率低于 30%时，CloudSearch 会为搜索域自动移除多余的搜索实例。

例如，某个应用需要三个分区，搜索域包含三个搜索实例（每个分区对应一个实例）。如果搜索请求超过每个搜索实例的处理能力，CloudSearch 会添加三个新的实例，并把分区拷贝到新增加的实例上，随着流量增长，还可以添加更多的实例，如图 6-1 所示。用户可通过 AWS 网站的 Account Activity 页面、AWS 管理控制台、命令行工具或 API 来查看 CloudSearch 所使用的资源情况。

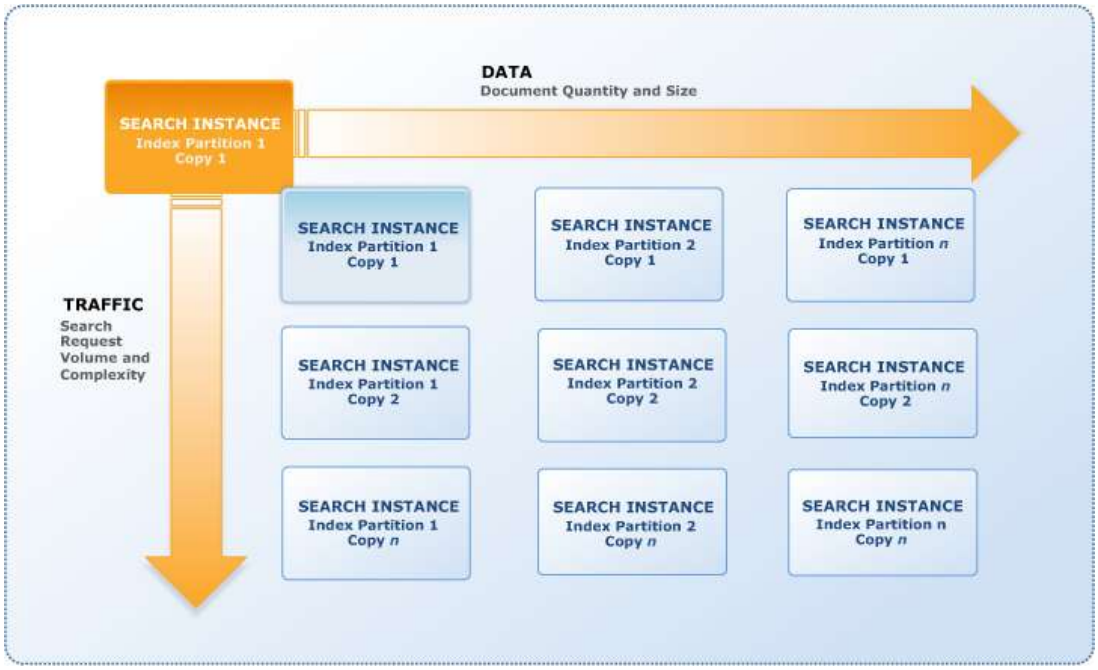

图 6-1　CloudSearch 搜索实例自动扩展

每个搜索实例类型能支撑的数据量主要依赖于文档（即搜索源数据）集合的大小以及索引字段的配置情况。下面我们将使用一个 Wikipedia 的样例文档和配置作为 benchmark，来说明每个搜索实例类型的能力。

CloudSearch 中，文档使用 Search Data Format（SDF）来描述，某个 Wikipedia 样例文档的 JSON 版本如代码清单 6-1 所示，文档大小约为 1 KB。

代码清单 6-1　Wikipedia 样例文档的 JSON 描述

```
{ "type": "add",
  "id": " wikipedia26678",
  "version": 5465249,
  "lang": "en",
  "fields": {
     "title": "Star Wars",
     "url": "http://en.wikipedia.org/wiki/Star_Wars",
     "author": "Jedi94",
     "type": "Article",
     "year": "1977",
     "teaser": "The Star Wars title card/logo, as seen in all films.
       'Star Wars' is an American epic space opera film series created by
       George Lucas. The first film in the series was originally released
       on May 25, 1977, under the title Star Wars, by 20th Century Fox,
       and became a worldwide pop culture phenomenon, followed by two
       sequels, released at three-year intervals. Sixteen years after the
       release of the trilogy's final film, the first in a new prequel
       trilogy of films was released. The three films were ..."
```

```
  }
}
```

样例文档中的每个字段都需要配置若干索引选项，如字段类型、是否可搜索、是否可分面、结果可否排序等。从文档数量的角度来看，每个选项都直接影响搜索实例的能力。上例中 Wikipedia 数据集的索引字段配置例子如表 6-1 所示。

表 6-1　某个 Wikipedia 文档的索引字段配置例子

字段名	类型	可否搜索	可否分面	结果可否排序
title	text	✔	✗	✗
url	text	✔	✗	✗
author	text	✔	✗	✗
year	uint	✗	✔	✔
type	literal	✔	✔	✗
teaser	text	✔	✗	✗

基于上述例子中的文档大小（1 KB）和索引配置情况，每种搜索实例类型可以处理的文档数目如表 6-2 所示。

表 6-2　每种搜索实例类型可处理的文档数

搜索实例类型	数据处理能力
小型搜索实例	1 百万文档
大型搜索实例	4 百万文档
超大型搜索实例	8 百万文档

以上只是一个示例，不同的文档大小和配置对实例能处理的文档数影响巨大。CloudSearch 最多可自动扩展到 10 个超大型搜索实例，能支持千万甚至亿级的文档数，如果用户有更高的需求，需向 Amazon 额外申请。

2) CloudSearch 架构

CloudSearch 架构图如图 6-2 所示。

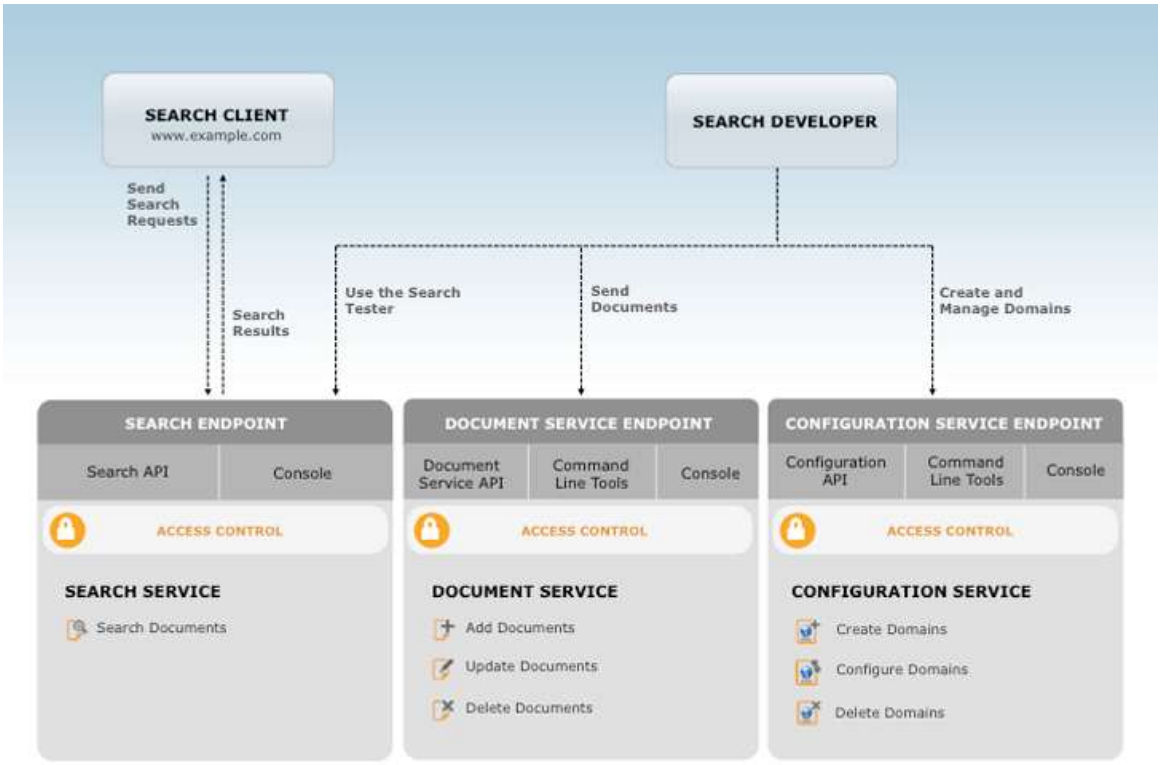

图 6-2　CloudSearch 架构图

图中显示 Amazon CloudSearch 提供了三种子服务：

- 配置服务

  配置服务使用户可以创建并配置搜索域，每个域封装了用于搜索的数据集。提供域名即可创建一个新的搜索域，然后，可通过指定索引选项、文本选项和分级表达式来配置搜索域。

  - 索引选项指定要包含在索引中的字段。可使用 AWS 管理控制台或命令行工具查看数据以自动配置默认的索引选项。
  - 文本选项指定搜索域特有的字典，以便在建立索引时忽略某些单词；文本选项还定义了术语的常用同义词，并把单词的多种变化映射到同一词根上，使得该词根可以匹配所有的单词变种。
  - 分级表达式是数学函数，可用来改变如何分级搜索结果。默认情况下，文档按一个文本相关度分数来分级，其考虑的是文档中搜索词的相似度及其频度。用户可在分级表达式中包含其它因子。

- 文档服务

  文档服务用于更改域中的可搜索数据。每个搜索域有一个唯一的文档服务 HTTP endpoint，当数据上传到域中时，会近似实时地自动建立索引并搜索就绪。

  按 SDF 格式描述的数据方可被搜索域接受。SDF 中，每个能作为搜索结果返回的项都代表一个文档，每个文档都有一个唯一的 ID（docid）、版本号、一个或多个字段（包含可用于搜索和结果返回的数据），文档字段可以包含任意 UTF-8 类型的字符数据。搜索域的索引选项指定了在索引中 SDF 文档字段之间的映射。

- 搜索服务

搜索服务处理搜索域的搜索请求。每个域都有一个唯一的 HTTP endpoint，当用户发送一条搜索请求，搜索服务返回一组按相关度排序的文档，搜索结果可以是 JSON 或 XML 格式。

CloudSearch 提供了丰富的查询功能，如在若干特定字段内搜索、执行复杂的布尔搜索、获取分面信息、指定返回结果中要包含什么数据等。

可使用 CloudSearch 控制台的 search tester 来测试查询语句。

6.2 Amazon Simple Workflow Service (SWF)

6.2.1 简介

Amazon 简单工作流服务（Simple Workflow Service，SWF）是构建可扩展与弹性应用的工作流服务[73]。在传统的开发方法中，如果应用处理的步骤运行在不同时间且运行时长不同，则保证它们可靠且无重复的执行是一件耗时费力的事情。如果应用处于分布式系统环境，各处理步骤之间的协调更是一种挑战。

在 Amazon SWF 中，开发者将应用中的各种处理步骤组织为“任务（task）”，SWF 负责协调这些任务以可靠且可扩展的方式运行。工作流是一组按特定顺序（有时候包含条件流或循环操作）执行的任务。工作流的每一次执行，都被视为一个独特的工作流执行（workflow execution）。

SWF 基于开发者的应用逻辑，来管理任务执行的依赖关系、执行计划与更新。SWF 存储任务、将任务分发给应用模块、跟踪任务进度并确保任务处于最新状态。SWF 是一个完全被管服务，用户无需操心软硬件管理、扩展、调优、打补丁或更新的事情，用户只需在 EC2 实例或任何具有网络访问的机器上使用任何语言调用 API 来访问 SWF 服务即可。

6.2.2 功能

在用户应用中，Amazon SWF 作为所有模块的协调中心来管理工作流，其功能包括：

- 维护应用状态
- 跟踪工作流的执行，并将进度记入日志
- 控制并分发任务
- 决定任务在哪个应用主机上执行

Amazon SWF 的使用也很简单[74]，步骤为：

1) 使用 AWS 管理控制台或 Amazon SWF API 来指定工作流的名字；
2) 使用 SWF API 启动一个新的工作流，即把一组工作流任务按特定的顺序组织起来（称为一个工作流执行）并开始执行；
3) 从 worker 机器（即应用中处理特定任务的某一模块）上调用 SWF API 来设定任务的顺序、管理条件流，以及执行某个工作流执行中的循环操作；
4) 从 worker 机器上调用 SWF API 来请求并执行工作流任务；
5) 在 AWS 管理控制台监控工作流执行的状态与进度，以及工作流执行中的任务情况。

6.2.3 特点

Amazon SWF 具有如下特点：

- 简单：作为一个 Web 服务，SWF 可以替代用户自身的复杂工作流方案及过程自动化软件，采用 SWF，用户无需操心过程自动化的底层管理，而只需关注应用的功能本身。
- 可扩展：SWF 可根据应用的使用情况无缝扩展。当添加更多工作流到应用中，或者增加工作流的复杂度时，都不会产生任何手动的工作流管理工作。
- 灵活性：用户可采用任何编程语言在 cloud 上或本地调用 SWF 来编写应用模块及协调逻辑。

6.2.4 定价

SWF 的费用与工作流执行、任务、数据传输量有关[73]。

6.2.5 产品细节

本节仅介绍若干 SWF API，更多产品细节请参考 SWF 文档[74]。Amazon SWF 提供了一组简单的 Web Service 接口来管理工作流：

- *StartWorkflowExecution*：启动一组按顺序排列的工作流任务，并将第一个任务指定给某个应用主机执行。
- *DescribeWorkflowExecution*：提供工作流执行与任务的状态。
- *PollForActivityTask*：应用主机（Cloud 上或本地）按连续循环的顺序请求并执行工作流任务。
- *RespondActivityTaskCompleted*：应用主机告知 SWF，任务已成功完成。之后，SWF 将下一个任务指定给应用主机，以继续工作流执行。
- *TerminateWorkflowExecution*：停止某个工作流执行，SWF 将不再指定该工作流执行中的任何任务给应用主机。

6.3 Amazon Simple Queue Service (SQS)

6.3.1 简介

Amazon 简单队列服务（Simple Queue Service，SQS）[8]为计算机之间交互消息的存储提供了一个可靠且高扩展的队列。开发者的应用可由多个执行不同任务的分布式组件构成，基于 Amazon SQS，数据可在这些组件之间移动，既不会丢失消息，也不要求每个组件都一直可用。SQS 与 EC2 及其它 AWS 服务紧密结合，使自动化工作流的构建变得容易。

Amazon SQS 将 Amazon 的 Web 扩展的消息框架对外封装为一种 Web 服务，Internet 上的任何一台机器都可以添加或读取消息，无需安装任何软件或进行特殊的防火墙配置。使用 SQS 的应用的各个组件可独立运行，无需使用同一技术开发，无需在同一网络运行，亦无

需同时运行这些组件。

SQS 可与 EC2、S3、SimpleDB 等一起使用，让应用更具灵活性和可扩展能力。比如一个常用场景：要创建一个集成且自动化的工作流，其中多个组件或模块需要相互通信，但它们无法同时处理同样数量的工作。在这种情况下，可以用 SQS 队列传送消息，并由 EC2 实例上运行的用户应用按顺序处理这些消息。EC2 实例读取队列、处理 job、然后将结果作为消息发布到其它 SQS 队列（另一应用可能需要做进一步的处理）。由于 EC2 可以动态向上或向下扩展应用，因此开发者可以根据 SQS 队列的情况轻松地改变计算实例的数量。

下面我们来看一个视频转码网站是如何集成使用 EC2、SQS、S3 和 SimpleDB 服务的。终端用户将需要转码的视频提交到该网站，网站将视频文件存储在 S3 上，并将一条消息（令为“请求转码消息”）加入 SQS 队列（令为“入口队列”），消息中的指针指向视频文件及目标视频格式。转码引擎运行在一组 EC2 实例上，从入口队列中读取请求转码消息，使用消息中的指针从 S3 获取视频文件，并将其转码为目标格式。转码后的视频文件存回 S3，并生成另一条消息（令为“回应消息”）加入另一个 SQS 队列（令为“出口队列”），消息中的指针指向转码后的视频文件。同时，视频文件的元数据（如格式、创建日期及长度等）可存储在 SimpleDB 中，并建立索引以便于查询。在这整个工作流中，一个专门的 EC2 实例持续监控入口队列，并根据入口队列中的消息数量来动态调整转码引擎所需要的 EC2 实例数目。

6.3.2 特点

Amazon SQS 具有如下特点：

- 可靠性：SQS 运行在 Amazon 高可用数据中心内，所有消息都在多台服务器和数据中心冗余备份，保证了消息不会丢失或不可用。
- 简单：开发者基于 5 个 API 就可以开始使用 SQS，即 *CreateQueue*、*SendMessage*、*ReceiveMessage*、*ChangeMessageVisibility* 及 *DeleteMessage*，每个 API 调用视为一次请求。SQS 还有更多 API 提供高级的功能，详见第 6.3.4 节。
- 可扩展：SQS 的设计目标就是允许任意数量的计算机在任意时刻读写任意数量的消息。
- 安全：SQS 的认证机制保证了队列中的消息不会被未经授权地访问。
- 廉价：通过 SQS 发送消息只需按请求数和数据传输量付费。

6.3.3 定价

每个 SQS 请求 0.000001 美元，数据传输量费用依 Region、入口/出口不同而稍有不同[8]。

6.3.4 产品细节

开发者可以创建无穷多个 SQS 队列，每个队列可有无穷多条消息。队列可在 7 个 Region 中创建，分别为美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）、欧洲（爱尔兰）、亚太（新加坡）、亚太（日本）和南美洲（圣保罗）Region。消息体最多可包含 64 KB 的任意格式的文本，队列中的消息最长可保留 14 天，可同时发送并读取消息。

当收到一条消息并进行处理时，该消息置于被锁状态（locked），以防止其它应用同时

处理该消息。如果消息处理失败，锁将会过期，消息恢复可用状态。如果应用需要更多时间来处理消息，可使用 *ChangeMessageVisibility* 操作来动态更改锁的超时设置。SQS 队列可与其它 AWS 账户匿名共享，也可在指定时间内共享给指定的 IP 地址访问。

SQS 提供了如下队列请求：

- *CreateQueue*：创建队列，可使用 AWS 账号访问该队列。
- *ListQueues*：列出现有的队列。
- *DeleteQueue*：删除某个队列。
- *SendMessage*：向某个队列添加消息。
- *SendMessageBatch*：向某个队列添加多条消息。
- *ReceiveMessage*：从某个队列获取一条或多条消息。
- *ChangeMessageVisibility*：更改某条已收到消息的可见超时值。
- *ChangeMessageVisibilityBatch*：更改多条已收到消息的可见超时值。
- *DeleteMessage*：从某个队列删除一条消息。
- *DeleteMessageBatch*：从某个队列删除多条消息。
- *SetQueueAttributes*：控制队列设置，如消息加锁时长（使消息被读取后，若干时间内不能再读）。
- *GetQueueAttributes*：获取队列信息，如队列中的消息条数。
- *GetQueueUrl*：获取队列 URL。
- *AddPermission*：将指定队列共享给另一个 AWS 账户。
- *RemovePermission*：删除某个 AWS 账户对某个队列的共享。

SQS 队列中消息的生命周期描述如下：

1) 应用系统使用 *SendMessage* 发送一条新消息到某个 SQS 队列；
2) 另一个应用系统调用 *ReceiveMessage* 来获取步骤 1)中加入的消息，并进行处理；
3) 当一条消息被 *ReceiveMessage* 调用返回后，在可见超时前，都不会被其它 *ReceiveMessage* 调用返回，以避免多个应用同时处理同一条消息。
4) 如果消息处理成功，应用系统调用 *DeleteMessage* 从队列中删除该条消息；如果系统处理消息失败，则在可见超时后，另一条 *ReceiveMessage* 调用可以重新读取该条消息。

6.4 Amazon Simple Notification Service (SNS)

6.4.1 简介

Amazon 简单通知服务（Simple Notification Service，SNS）[52]作为一个 Web Service，使得从 cloud 端设置、操作、发送通知变得容易，SNS 为开发者提供了一种高可扩展、灵活且经济的方式，实现了从某个应用发布消息，并立即将消息送达给订阅者或其它应用。

SNS 提供了简单的 Web Service 接口以及基于浏览器的管理控制台，可用来创建要通知给应用或人的话题（topic）、订阅话题、发布消息，以及按客户端协议（如 HTTP、email、SMS 等）传送消息。SNS 采用推（push）模式来传送通知，避免了拉（poll）模式的轮询操作。基于 SNS，无需复杂的中间件和应用管理模块，就可以构建高可靠性、事件驱动的工作流和消息应用。SNS 的应用场景包括监控应用、工作流系统、时间敏感的信息更新、移

动应用等。

目前 SNS 为 beta 版，一个 AWS 账户只能创建 100 个话题，若有更多需求，请联系 Amazon。

6.4.2 功能

SNS 的设计目标是满足事件驱动的应用需求，构建可无缝扩展的灵活、可靠、经济的消息解决方案。SNS 具有很高的定制度，能满足广泛的应用需求。

使用 SNS 相当简单，步骤如下：

1) 创建一个话题：话题是一个访问点，用来区别主题或事件类型，借助话题可以发布消息并允许客户端订阅通知。
2) 为话题设置策略：话题的所有者可以为话题设置策略，如谁可以发布消息或订阅通知，或支持何种通知协议（如 HTTP/HTTPS、email、SMS、SQS 等）。单个话题支持在多种传输协议上传送通知。
3) 为话题添加订阅者：订阅者为希望获取某个话题的通知的客户端，客户端可以主动订阅，也可被话题所有者添加到订阅列表。订阅者指定传递通知所用的协议格式及 endpoint（URL、email 地址、电话号码等）。收到订阅后，SNS 会向指定的 endpoint 发送确认消息，要求订阅者明确确认要接收来自某个话题的通知。
4) 发布消息/发出通知：当话题所有者有更新要通知订阅者时，可向话题发布消息，此将立刻触发 SNS 将消息分发给所有有效的订阅者。

6.4.3 特点

Amazon SNS 具有如下特点：

- 可靠：SNS 运行在 Amazon 的可信网络基础架构与数据中心中，所以话题随时可用。而且所有向 SNS 发布的消息都在多台服务器和数据中心中冗余备份，因而消息不会丢失。
- 可扩展：SNS 能支持应用在任意时刻发布任意数量的消息。
- 简单：大多数情况下，开发者只需 3 个 API 就可以开始使用 SNS：*CreateTopic*、*Subscribe* 及 *Publish*，其它 API 提供了更多高级功能。
- 灵活：基于 SNS，处于不同设备上的应用和终端用户可通过 HTTP/HTTPS、email/email-JSON、短信（Short Message Service，SMS）或 SQS 队列来接收通知。
- 安全：SNS 提供了访问控制机制，确保话题和消息不会被未经授权地访问。话题所有者可以设置策略，指定谁可以发布或订阅该话题，而且话题所有者还可以指定传输协议必须为 HTTPS 来加密通知。
- 经济：用户按请求数、发送通知数和数据传输量计费。
- 与其它 AWS 集成：SNS 可与包括 SQS、EC2 在内的其它 AWS 集成使用。例如，运行在 EC2 上的应用可以发布事件/信息更新给 SNS，SNS 立即将更新传递给其它应用或终端用户。此外，订阅者可以选择 SQS 作为分发协议，使通知发送到某个 SQS 队列上。未来，SNS 将与更多 AWS，如 S3 和 SimpleDB 等集成。所有的 AWS Region 都支持 SNS，但目前暂时只有美国东部 Region 支持短信通知方式。

6.4.4 定价

SNS 按 API 请求数、通知数及数据传输量计费[52]。

- 每月的头 100,000 个 SNS API 请求是免费的，此后每 100,000 个 SNS API 请求收费 0.06 美元。
- 通知数
  - HTTP/HTTPS：每月头 100,000 个 SNS HTTP/HTTPS 通知免的，此后每 100,000 个 HTTP/HTTPS 通知收费 0.06 美元；
  - Email/Email-JSON：每月头 1000 个 SNS Email/Email-JSON 通知免费，此后每 100,000 个 Email/Email-JSON 通知收费 2 美元；
  - 短信：每月头 100 条短信通知免费，此后每 100 条短信通知收费 0.75 美元。
  - SQS：SQS 通知免费。
- 按所在 Region 及入口和出口数据传输量收费。

6.4.5 常用场景

Amazon SNS 的常见应用场景很多，例如：

- 应用集成

  SNS 可用于工作流系统，在分布式应用中传递事件、在不同存储之间移动数据，或更新系统中的记录。例如，在一个订单处理应用中，每当一笔交易发生时，会产生一个通知消息；当客户下单、交易转到支付环节时，一个订单确认消息会发布到一个 SNS 话题上。在这个例子中，各个订阅者（商家、客户、供应链合作商）可通过 email 订阅该话题，SNS 发出的通知可以即时向所有订阅者更新支付处理情况。SNS 还可用于构建更鲁棒的订单处理系统中，通过 HTTP 发送的通知可以触发相关组件的实时处理，如库存系统或物流服务等。例如，用户下单后，库存管理系统可以收到订单通知；当订单包装完成时，物流服务可以收到通知。
- 时间敏感信息的更新

  SNS 可支持商业、社交网络或其它组织中对信息分发的需求。通知可用来提醒订阅者某个关心的事件，订阅者可使用不同的设备来收取订阅的通知。例如，假设某个社交社区紧密关注某项运动或队伍，比分的更新、赛程、队员、事件及可用门票等，这些实时信息可以轻松地推送给感兴趣的成员。SNS 提供了一种弹性低成本的方式实现大范围内参与者的实时沟通，并可扩展到任意所需的容量。
- 移动应用

SNS 可用于将时间关键的应用事件传送给移动应用和设备。移动应用越来越多地用于从各种源（包括天气、交通、股票、体育，甚至多人游戏）分发并集成实时信息。开发者可以很容易地将移动应用与 SNS 集成，通过 HTTP、邮件及短信等方式发送消息并接收通知。

6.4.6 产品细节

1) 使用 SNS

用户可通过 AWS 管理控制台访问 SNS，也可通过 API[75]调用 SNS，最常用的 API 及其功能描述如下：

- CreateTopic：创建一个话题，创建者提供话题名。有了话题这个访问点，订阅者才能注册通知，发布者才能发送消息。
- Subscribe：将一个新订阅者注册到某个话题。调用该 API 时要指定话题、传输协议（HTTP/HTTPS、Email/Email-JSON、短信、SQS）及 endpoint（URL、Email 地址、电话号码、SQS 队列）。在订阅者明确确认要接收某个话题的通知之前，SNS 不会分发任何通知。
- Publish：将消息发布给某个话题，并随之将该消息分发给该话题的所有订阅者。调用该 API 时要指定话题，并提供消息内容。SNS 目前允许消息内容最大为 32 KB。

2) 与 SQS 集成

SQS 和 SNS 都是 AWS 中的消息服务，但两者提供不同的功能。SQS 是一个供分布式应用使用的消息队列服务，分布式模块之间通过拉模式来交换消息，使得消息发送和接收组件之间解耦，各组件无需同时可用。SNS 使应用能通过推模式将时间关键的消息发送给多个订阅者，而无需定期轮询的拉模式。SQS 和 SNS 结合使用，可使某一事件发生时，能立刻向应用发送通知消息，同时，消息可持久保存在 SQS 队列中，供之后其它应用处理。

6.5 Amazon Simple Email Service (SES)

6.5.1 简介

Amazon 简单邮件服务（Simple Email Service，SES）[76]面向企业和开发人员提供了高扩展、低成本的事务级电子邮件发送服务，SES 省去了自己构建市场级电子邮件解决方案的复杂和开销，也避免了采用第三方邮件服务而要关注的许可、安装和运行工作。SES 和其它 AWS 服务实现了集成，使得从 AWS 服务上（如 EC2）的应用发送电子邮件变得简单。

构建大型的市场级、事务级电子邮件解决方案通常是一件复杂而昂贵的事情。为了优化成功发送邮件的百分比，需要考虑诸多问题，如邮件服务器的管理、网络配置、符合 ISP（Internet Service Provider）对电子邮件内容的严格标准等。此外，大量第三方的电子邮件解决方案需要进行合同和价格协商，成本开销也颇为可观。

Amazon SES 基于 Amazon.com 在大规模电子邮件体系架构上的多年经验，使以上问题迎刃而解。企业用户可通过 SMTP 或 API 调用访问 SES，向成千上万的终端客户发送电子邮件。SES 使用内容过滤技术扫描发送的电子邮件，以确保电子邮件内容符合 ISP 的要求，扫描通过的电子邮件被加入发送队列，扫描失败的电子邮件被退回发送者做相应处理。为了帮助企业用户进一步提高其与客户的电子邮件通信质量，SES 提供了一个内置的反馈功能，包括退信通知、发送失败后的尝试次数、垃圾邮件投诉等。

6.5.2 功能

Amazon SES 使企业和开发人员能快速、低成本地向大量客户发送事务级的电子邮件，SES 的使用步骤如下：

- 注册：完成注册后，可访问 Amazon SES 沙箱，SES 沙箱是专门为开发人员测试和评估电子邮件服务而设计的环境。

- 核对域或电子邮件地址：在使用 SES 发送电子邮件之前，应先到 AWS 管理控制台执行验证流程，确保用来发送电子邮件的域或地址正确。
- 切入生产环境：完成准备工作后，即可请求访问生产环境，从而从沙箱环境切换到生产环境，并可向客户发送电子邮件。申请访问生产环境只需几分钟时间，通常会在 24 小时内收到响应结果。
- 发送电子邮件：使用 SMTP 或 API 将电子邮件消息加入发送队列。
- 获取反馈：SES 提供了电子邮件发送的统计信息，可使用 AWS 管理控制台或查询语句，快速获取电子邮件发送、退回及投诉等关键统计数据。

送达率（deliverability）代表电子邮件成功发送到预期目的地的概率。一个不幸的事实是，全球电子邮件流量中的很大一部分是垃圾邮件。ISP 部署了自动过滤器来检测电子邮件消息是否含有疑似垃圾邮件的内容，并阻止投递这些消息。在 Internet 上发出的电子邮件信息大部分都被一个或多个 ISP 扫描过，每个 ISP 均有自己的垃圾邮件过滤器。但垃圾邮件过滤器并不完美，即使是合法的电子邮件，有时候仍会被错误地识别为垃圾邮件，从而被阻止传递。

在决定是否传递一封电子邮件时，ISP 考虑的一个因素是该电子邮件的发送来源，如果发送者 IP 地址或域有发送垃圾邮件的历史记录，则 ISP 可能阻止来自该 IP 地址或/和域的所有电子邮件。

Amazon SES 主动采取保障措施以防止有问题的内容被发送出去，因此 ISP 一直会收到高质量的电子邮件，从而将 SES 服务视为可信的电子邮件源，这使得 SES 所有用户的发送率最大化并可信任。SES 所采取的一些保障措施如下：

- ISP 通常会将电子邮件流量的突然增加解读为潜在的垃圾邮件征兆，并可能阻止这类电子邮件。为避免这种危险，SES 自动平滑增加电子邮件流量直到用户的目标流量值（详见第 6.5.5 节）。
- SES 使用内容过滤技术检测并阻止含有垃圾邮件或恶意软件的消息被发送出去。
- SES 维护来自主要的 ISP 的投诉反馈表。投诉反馈表显示了哪些电子邮件的接收者将该邮件标记为垃圾邮件。SES 提供这些信息帮助用户改进发送策略。

除了 SES 所采取的以上措施外，用户也可使用 Easy DKIM（Domain Keys Indentified Mail，域名密钥识别邮件标准）来配置邮件的发送，此时，SES 将代表用户使用 DKIM 加密邮件。大多数 ISP 都接受这种工业级标准的方法，可以使用户的邮件发送声誉与所在域保持一致。

6.5.3 特点

Amazon SES 具有如下特点：

- 简单：使用 SMTP 或 API 调用即可通过 SES 发送电子邮件，省却了构建维护电子邮件解决方案的大量工作，而且还可通过 SES 来监控电子邮件发送活动与送达率统计信息。
- 经济：用于仅为发送的电子邮件数及数据传输量付费。
- 可靠：SES 运行在 Amazon 的可信网络基础架构及数据中心上，所有发出的电子邮件都会在多台服务器和数据中心内冗余备份，确保数据的高可用性与持久性。
- 可扩展：SES 基于的扩展技术亦被用于 Amazon 的全球网站，来实现每年发送数以十亿计的电子邮件。

- 与其它 AWS 的集成：用户可通过 Amazon SNS 来跟踪退信和投诉事件，也可通过 Amazon Route 53 来设置 Easy DKIM 或验证任何被管域，从 EC2 及 Elastic Beanstalk 发送电子邮件还可享受一定的免费使用优惠。

6.5.4 定价

与发送的电子邮件数、发送与接收的数据量、附件大小等有关[76]。

6.5.5 产品细节

1) 管理 Amazon SES

可通过 AWS 管理控制台或 API 来设置并管理 Amazon SES，包括查看发送限制、发送测试消息、设置退信与投诉通知、管理验证域及电子邮件地址、设置 Easy DKIM 等。

2) 将 SES 与其它 AWS 集成

SES 可与 EC2、Elastic Beanstalk、SNS 及 Route 53 等无缝集成。Amazon 推荐 SES 作为 EC2 发送电子邮件的解决方案，使用 AWS SDK 或 API 可以为运行在 EC2 实例上的应用增加电子邮件功能。

3) 电子邮件发送统计信息

SES 自动收集邮件发送的统计信息，这些信息可通过 API 调用实时获取，包括：

- 成功的发送尝试次数
- 拒绝的消息
- 退回数
- 投诉

4) 发送量

每个 SES 发送者都要受以下因素的限制[77]：

- 发送配额：表示 24 小时内最多能发送的电子邮件数；
- 最大发送速率：表示每秒能发送的最大电子邮件数；

获得 SES 的生产环境访问权限后，默认 24 小时可以发送 10,000 封电子邮件，如果用户发送邮件的质量一直保持高质量，该数值也会随之提高，可通过 AWS 管理控制台的 Amazon SES tab 页或调用 GetSendQuota API 来获取当前的发送配额值。

第七章 市场类（Marketplace）

7.1 Amazon Marketplace

AWS Marketplace[78]是一个在线的应用商店，帮助客户查找、购买并立即开始使用运行于 AWS 云上的软件，其中一些软件来自可信的提供商，如 SAP、Zend、Microsoft、IBM、Canonical、10gen，以及很多广泛使用的开源软件，如 Wordpress、Drupal、Mediawiki 等。开发者也可将自己开发的软件发布到 Amazon Marketplace 上，供 Amazon 用户购买。

Marketplace 和 Apple App Store 类似，阿里云的云应用市场（http://market.aliyun.com）也是同样的功能。请访问 https://aws.amazon.com/marketplace/ref=mkt_ste_pas 进一步了解 Amazon Marketplace，本文不多做阐述。

第八章 网络类（Networking）

8.1 Amazon Route 53

8.1.1 简介

Amazon Route 53 是一个高可用与高可扩展的域名系统（Domain Name System，DNS）的 Web 服务[79]。企业和开发人员可基于 Route 53 提供极其可靠且经济的 DNS 服务，既可有效地将用户请求连接到 AWS 上运行的应用上（如 EC2 实例、ELB、S3 bucket），也可将用户请求路由到 AWS 外部。

Route 53 的 DNS 服务器遍布全球网络，对某个域的查询会自动路由到最近的 DNS 服务器，从而实现低延迟的 DNS 查询应答。

8.1.2 功能

Route 53 对外提供了 Web Service 接口，易于使用，使用步骤为：

1) 在 Route 53 服务页面[79]点击 Sign-up 按钮，注册该服务；
2) 创建一个 hosted zone，存储用户自有域的 DNS 记录。创建了域后，会收到 4 个 Route 53 名字服务器地址，这 4 台服务器分处 4 个不同的 Top-Level Domain（TLD）；
3) 用户的 hosted zone 先要加入一些基本的 DNS 记录，包括第 2)步中的 4 个虚拟名字服务器，可使用 AWS 管理控制台或 *ChangeResourceRecordSet* API 来增删改 DNS 记录，Route 53 支持若干 DNS 记录类型[80]；
4) DNS 记录更新后，要通知 hosted zone 的注册者。

如果想把域从其它 DNS 服务迁移到 Route 53，需进行如下操作：

- 获取该域名下的 DNS 记录数据列表，一般是“区域文件（zone file）”形式，可从现有 DNS 提供商处获取。
- 按上述四步开始使用 Route 53。

8.1.3 特点

Route 53 具有如下特点：

- 高可用性与可靠性：Amazon DNS 服务器的分布式特性决定了 Route 53 的高可用性与可靠性，并有 SLA[81]予以保证。
- 扩展性：当查询量大幅上升时，Route 53 会不停服务地自动扩展。
- 与其它 AWS 集成：Route 53 可用来将域名映射到 EC2 实例、S3 bucket、CloudFront 分发包或其它 AWS 资源。通过和 Amazon IAM 集成，可以精细控制谁能更新 DNS 数据。

- 简单：可通过 AWS 管理控制台或 API 配置 DNS 选项，也可以编程方式将 Route 53 集成到用户的 Web 应用中，例如，当用户创建一个新的 EC2 实例时，可使用 Route 53 的 API 创建一个对应的新 DNS 记录。
- 安全：通过将 Route 53 和 IAM 集成，可以为 AWS 账户下的每个用户赋予唯一的凭证及管理权限，指定谁对 Route 53 的哪一部分服务有访问权限。
- 灵活性：Route 53 提供了 Weighted Round-Robin（WRR），即所谓的 DNS 负载均衡，WRR 允许为 DNS 记录赋予权重值，以指定不同的 endpoint 上要路由多少流量。

8.1.4 定价

Route 53 的价格与 hosted zone、DNS 查询类型等有关[79]。

8.1.5 产品细节

之所以取名为 Route 53，是因为该服务的 DNS 服务器使用 53 端口作为 DNS 查询和响应端口。

1) Amazon Route 53 的 API

Route 53 提供了一组简单的 API，用于创建并管理 DNS 记录。最常用的 Route 53 API 及其功能如下：

- *CreateHostedZone*：创建一个新的 hosted zone，来存储用户 DNS 数据。创建完成后，用户会收到用于 DNS 服务的四个域名服务器地址。
- *GetHostedZone*：列出指定 hosted zone 的信息。
- *DeleteHostedZone*：删除一个 hosted zone。
- *ChangeResourceRecordSets*：加载并编辑 hosted zone 中的 DNS 资源记录。
- *ListResourceRecordSets*：获取 hosted zone 中全部或按记录名及类型筛选后的资源记录集。

2) Route 53 的全球网络

Route 53 在全球部署有 DNS 服务器网络，部署地包括：

美国：弗吉尼亚州 Ashburn（2 台）、得克萨斯州达拉斯/Fort Worth（2 台）、佛罗里达州 Jacksonville、佛罗里达州迈阿密、加利福尼亚州洛杉矶（2 台）、加利福尼亚州 Palo Alto、加利福尼亚州 San Jose、纽约州纽约（2 台）、新泽西州 Newark、华盛顿州西雅图、印第安纳州 South Bend、密苏里州圣路易斯。

欧洲：阿姆斯特丹（2 台）、都柏林、法兰克福（2 台）、伦敦（2 台）、米兰、巴黎（2 台）、斯德哥尔摩。

亚洲：香港、大阪、新加坡（2 台）、悉尼、东京。

南美洲：圣保罗。

8.2 Amazon Virtual Private Cloud (VPC)

8.2.1 简介

Amazon 虚拟私有云（Virtual Private Cloud，VPC）[11]可部署一个私有的、隔离的 AWS 云，用户可在该 VPC 中的虚拟网络上启动 AWS 资源。用户可在 VPC 上定义一个非常类似传统网络的虚拟网络拓扑，且对该虚拟网络环境拥有完全的控制权，包括选择 IP 地址范围、创建子网、配置路由表及网关等。

用户可以轻松定制 VPC 的网络配置，例如，为 Web 服务器创建一个能访问 Internet 的开放子网，而将包括数据库、应用服务器在内的后端系统放在不能访问 Internet 的私有子网中，还可利用多层次的安全机制（包括安全组和网络访问控制列表）来控制对各个子网内 EC2 实例的访问。

此外，用户也可创建硬件虚拟私有网络（Virtual Private Network，VPN）连接公司内部的数据中心与 VPC，使 AWS 云成为公司数据中心的一个扩展。

8.2.2 功能

基于 Amazon VPC，能实现如下功能：

- 在 AWS 可扩展的基础架构上创建 Amazon VPC，并指定任意的私有 IP 地址范围。
- 根据 VPC 上所运行的应用和服务的需要，将 VPC 的私有 IP 地址范围划分为一个或多个公开或私有的子网。
- 使用网络访问控制列表控制各个子网的入口和出口访问。
- 在 S3 上存储数据，并设置权限，如仅能从 VPC 内访问数据。
- 为 VPC 中的实例指定多个 IP 地址并绑定多个弹性网络接口。
- 将一个或多个 Amazon 弹性 IP 地址绑定到 VPC 的任意实例上，使其能从 Internet 上直接访问到。
- 使用加密 VPN 连接 VPC 和用户自己的 IT 设施，并将现有的安全和管理策略扩展到 VPC 实例上。

8.2.3 特点

Amazon VPC 具有如下特点：

- 多种连接选项

  根据开放或保密 AWS 资源的需求，可将 VPC 连接到 Internet 或用户数据中心。
  - 直接连接到 Internet（开放子网）：在开放子网中启动实例，使其能收发 Internet 流量。
  - 使用网络地址转换（Network Address Translation，NAT）连接到 Internet（私网）：私网用于不想将实例直接暴露给 Internet 的情形。实例运行在私有子网，使用私有 IP 地址，并通过开放子网上的 NAT 实例来路由收发请求以访问 Internet。
  - 安全连接至公司数据中心：所有 VPC 实例上的出入流量都可通过工业级标准的加密 IPsec 硬件 VPN 连接而路由到用户公司的数据中心上。
  - 组合连接：按照实际需要，只需配置 VPC 的路由表，即可将 VPC 连接到 Internet 或用户公司的数据中心。
- 安全：VPC 提供高级安全特性，如安全组、网络访问控制列表，可在实例和子网

级别过滤出口和入口流量。数据可存储在 S3 上，以保证仅能从 VPC 内的实例访问数据。此外，如果有特殊的隔离要求，也可考虑在专有的硬件上启动专有实例类型。

- 简单：使用 AWS 管理控制台即可简便快速地创建 VPC，Amazon 提供了常用的网络设置，用户只需挑选合适的选项，点击“Create my VPC”即可。AWS 会自动创建子网、IP 地址范围、路由表及安全组等。
- 具备 AWS 服务所共有的可扩展和可靠性。

8.2.4 定价

除了使用 EC2 实例付费外，使用 VPN 时，每个 VPN 连接每小时收取 0.05 美元的费用。

8.2.5 常用场景

Amazon VPC 可用于多种应用场景，例如：

- 面向公众的简单网站：可在 VPC 中运行一个简单的 Web 应用，如博客或网站等，并通过创建安全组规则来保护网站，使 Web 服务器只对来自 Interent 的 HTTP 和 SSL 入口请求进行响应，同时禁止 Web 服务器建立到 Internet 的出口连接。AWS 管理控制台的 VPC 创建向导中的“VPC with a Public Subnet Only”选项即可满足本应用场景。
- 多层 Web 应用：可在 VPC 中运行多层 Web 应用，并在 Web 服务器、应用服务器和数据库之间设置严格的访问的安全限制。例如，在开放子网中启动 Web 服务器，但把应用服务器和数据库部署在私有子网中，从 Interent 无法直接访问应用服务器和数据库，但可通过 NAT 实例来访问 Internet，执行补丁下载等操作。用户可使用网络访问控制列表及安全组提供的出入口包过滤机制，来控制服务器和子网之间的访问。AWS 管理控制台的 VPC 创建向导中的“VPC with Public and Private Subnets”选项即可满足本应用场景。
- 连接到用户数据中心的可扩展 Web 应用：用户可创建一个 VPC，诸如 Web 服务器之类的实例运行在一个子网中与 Interent 通信，诸如应用服务器之类的实例运行在另一个子网中，与用户公司网络中的数据库通信。在 VPC 和用户公司网络之间以 IPsec VPC 连接，以保证应用服务器和数据库通信的安全。运行在 VPC 中的 Web 服务器和应用服务器可以利用 EC2 的弹性及 Auto Scaling 的特性来动态向上或向下扩展。AWS 管理控制台的 VPC 创建向导中的“VPC with Public and Private Subnets and Hardware VPN Access”选项即可满足本应用场景。
- 将用户公司网络扩展至云上：将用户公司网络和 VPC 连接起来后，可以把用户公司的应用迁移到云上，启动额外的 Web 服务器、或添加更多的计算资源到用户网络中。由于 VPC 可以启动在用户公司防火墙后，因此可以在不改变终端客户如何访问应用的前提下，将用户的 IT 资源无缝地迁移到云端。AWS 管理控制台的 VPC 创建向导中的“VPC with a Private Subnet Only and Hardware VPN Access”选项即可满足本应用场景。
- 灾难恢复：用户可借助 Amazon EBS 卷将关键数据从自己的数据中心周期性地备份到若干 EC2 实例上，或将自己的虚拟机映像文件导入到 EC2。当用户自己的数据中心遇到灾难事件，用户可在 AWS 中快速启动替代的计算资源以保障业务的连续性。当灾难结束后，可把业务关键数据传回自己的数据中心，并在不需要时中止

EC2 实例的运行。

8.2.6 产品细节

1) 使用 Amazon VPC

先注册 Amazon EC2 服务，完成后，到 AWS 管理控制台的 Amazon VPC tab 页，点击“Get Started Creating a VPC”按钮，在系统提供了四种基本网络拓扑中选择最合适自己的一个，然后点击“Create VPC”按钮。VPC 创建完成后，即可在 VPC 中启动 EC2 实例了。

2) 与其它 AWS 资源集成

目前，可在 VPC 中使用如下 AWS 资源和特性：运行 Linux/Unix 或 Windows 的 EC2 实例、用于持久化块存储的 EBS 卷、Auto Scaling、EC2 VM 导入、使用 CloudWatch 监控 EC2 实例的资源利用率。

通过 Internet 网关，还可从 VPC 访问 S3 及其它 AWS。借助 Amazon IAM 的安全策略及 EC2 的安全组，可以限制只有绑定到 VPC 的弹性 IP 地址可以访问 AWS 资源，如 S3 bucket、SimpleDB 域、SNS 话题、SQS 队列等。

3) 注意事项

- 目前，VPC 暂不支持使用 AWS Elastic Beanstalk、ElastiCache、支持 Microsoft SQL Server 的 RDS；
- 目前，VPC 暂不支持 EC2 CC1、CG1 及微型实例类型；
- VPC 不支持 Amazon DevPay 支付的 AMI；
- VPC 还有如下限制，若需求超过此限制，可向 Amazon 提交申请[82]：
  - 每个 AWS 账号在每个 Region 至多可创建 5 个 VPC；
  - 每个 VPC 至多可创建 20 个子网；
  - 每个 AWS 账户在每个 Region 至多可拥有 5 个 VPC 弹性 IP 地址；
  - 每个 VPC 至多可创建 10 个 Hardware VPN 连接；
  - 更多限制请参考 VPC 用户指南[83]。

8.3 AWS Direct Connect

8.3.1 简介

AWS 直连（Direct Connect）[84]可在用户个人的 IT 设施（如数据中心、办公室、托管主机等）和 AWS 之间建立私人专属的网络连接，以降低网络开销、增加带宽出口量，并获得更佳稳定可靠的 Internet 连接。

AWS 直连采用标准的 802.1q VLAN 协议，将专属的网络连接划分为多个虚拟接口，这样基于同样的连接，既可以公网 IP 地址访问公共资源（如 S3 上存储的对象），也可用私有 IP 地址访问私有资源（如运行在 VPC 中的 EC2 实例），而且公有和私有网络环境是相互隔离的，用户可在任何时候重新配置虚拟接口。

8.3.2 特点

AWS 直连具有如下特点：

- 降低带宽开销：用户数据直接进出 AWS，减少了中间 ISP 的开支。专属连接上的数据传输计费按 AWS 直连数据传输率，而不是按 Internet 数据传输率。
- 稳定的网络性能：AWS 直连较之 Internet 连接具有更稳定的网络性能，包括网络延迟等。
- 与其它 AWS 服务兼容：AWS 直连作为一个网络服务，可与所有需要访问 Internet 的 AWS 服务结合使用，例如 S3、EC2、VPC 等。
- 到 VPC 的私有连接：使用 AWS 直连可以建立从用户个人网络到 Amazon VPC 的私有虚拟接口，以在用户网络和 VPC 之间建立私有、高带宽的网络连接。通过多个虚拟接口，甚至可在用户网络和多个 VPC 之间建立私有连接，并保持各个连接之间的隔离。
- 弹性：AWS 直连提供 1 Gbps 和 10 Gbps 的连接，如果需要更高带宽，可以快速部署多个连接。AWS 直连还可在 Interent 和用户 VPC 之间建立 VPN 连接，虽然使用 VPN 硬件也可建立 VPN 连接，但其数据传输率不能持久保持在 4 Gbps 以上，而采用 AWS 直连就没有这样的问题。
- 简单：通过 AWS 管理控制台即可快速注册并使用 AWS 直连服务，管理控制台提供了一个统一界面来有限管理所有的直连及虚拟接口。在配置了一个或多个虚拟接口后，也可为网络设备下载定制的路由器模板。

8.3.3 定价

AWS 直连服务的费用主要来自于直连端口和数据传输量两方面[84]。直连端口按每端口每小时收费，1 Gbps 端口每小时 0.30 美元，10 Gbps 端口每小时 2.25 美元。入口数据量免费，出口数据量依 Region 不同而略有不同，大致在每 GB 0.03 美元左右。

8.3.4 产品细节

1) AWS 直连的常用场景

- 大数据集处理：用户可使用 AWS 直连将业务关键数据从自己的数据中心、办公室或托管主机直接传输到 AWS 或反之，绕过 ISP，也不会有网络拥塞的困扰。
- 实时数据推送：AWS 直连适合需要推送实时数据的应用，例如，音频和视频应用需要稳定的网络延迟，但 Interent 上网络延迟是变化的，而基于 AWS 直连，用户可以控制数据如何路由，以保证更稳定的网络性能。
- 混合环境：AWS 直连可构建混合网络环境，以满足需要专属连接的管理要求。用户可在自己已有的网络架构之上，基于 AWS 直连建立私有专属连接，从而构建混合型网络环境。

2) AWS 直连的位置

AWS 直连在全球 8 个地区可用，表 8-1 列出了 AWS 直连与 AWS Region 的互联关系。Amazon 今年（2012）正在规划更多的全球 AWS 直连点。

表 8-1　AWS 直连位置与 Region 的互联关系

AWS 直连位置	AWS Region
CoreSite 32 Avenue of the Americas, NY	US East (Virginia)
CoreSite One Wilshire & 900 North Alameda, LA	US West (Northern California)
Equinix DC1 - DC6 & DC10	US East (Virginia)
Equinix SV1 & SV5	US West (Northern California)
Equinix SG2	Asia Pacific (Singapore)
Equinix TY2	Asia Pacific (Tokyo)
TelecityGroup, London Docklands'	EU West (Ireland)
Terremark NAP do Brasil	South America (Sao Paulo)

3) 支持 AWS 直连的 APN 合作者

如果用户现有的 IT 设施不处于以上 AWS 直连点，可寻求 AWS 的 APN（AWS Partner Network，AWS 合作者网络）技术和咨询合作者（以前称为 AWS 直连解决方案提供商）[85] 的帮助来使用 AWS 直连。APN 合作者会帮助建立 AWS 直连点和用户数据中心、办公室或托管主机之间的网络链路，也能帮助架构混合的网络环境。

4) 开始使用 AWS 直连

- 选择 AWS 直连点、要使用多少条连接及端口大小（1 Gbps 或 10 Gbps），可同时使用多个端口来提高带宽或冗余。
- 使用 AWS 管理控制台创建连接请求。
- 如果用户要从 AWS 直连点之外接入，可联系 AWS 代理。
- 确认请求后，用户会收到一封邮件，包含授权信与连接设备信息（Letter of Authorization-Connecting Facility Assignment，LOA-CFA）[86]。
- 将 LOA-CFA 提供给 APN 合作者或用户自己的服务提供商，以帮助建立连接。
- 连接建立好后，使用 AWS 管理控制台来配置一个或多个虚拟接口来建立网络连接。

第九章 支付与计费类（Payments & Billing）

9.1 Amazon Flexible Payments Service (FPS)

9.1.1 简介

Amazon 弹性支付服务（Flexible Payment Service，FPS）[87]是第一个完全为开发者设计的支付服务，它架构在 Amazon 可靠且可扩展的支付框架之上，为开发者提供了方便的方法向成千上万的 Amazon 客户收取费用（当然要取得客户的事先许可），而这些 Amazon 客户可以使用已经在 Amazon 上使用的支付信息（如登录凭证、发货地址等）来支付。

开发者在其网站上进行售卖商品或服务等操作时，可集成 Amazon FPS 以提供支付功能。Amazon FPS 提供了无与伦比的灵活性，开发者可构建支付流程，包括用于多事务交易的标准流程。这些流程允许资金的支付方与收讫方对金钱的流动附加条件和限制，例如，支付方可设置每周向某个收讫方转账的限额，也只有该收讫方可以退还资金，且限额与每周该支付方支付的限额一致。

Amazon FPS 提供了易于集成的轻量级 API，且按用例分类成若干称为 Quick Starts 的交互包，FPS 亦有完善的文档、SDK 和样例支持。

9.1.2 功能

FPS 支持使用信用卡、银行账号及 Amazon 支付账号来支付或收讫资金。FPS 的 Quick Starts API 包实现了常用的支付交易类型，如一次性支付、按期支付、预付费等。FPS Quick Starts API 包由以下几部分组成：

- Quick Start 基本包[88]：用于售卖实物商品、电子产品、服务的一次性支付。
- Quick Start 高级包[89]：用于订阅、基于使用的服务（如数字音乐、在线存储）的周期性或后支付。
- Quick Start 应用市场包[90]：用于买方和第三方卖方之间的交易、削减交易、控制谁来支付交易流程费用等。
- Quick Start 累加支付[91]：累加多笔支付（包括微型支付操作）到一笔大型交易，并在服务之前或之后向客户收取费用。
- 开发者还可使用 Quick Start 账户管理包[92]以编程方式访问账户活动记录，也可使用 Amazon 支付网站[93]来查看账户活动记录与余额。

FPS 还提供了支付指南（Payment Instructions）功能来设置一笔交易的条件和限制，Quick Starts 提供了一组用户接口，可设置若干参数以支持大多数用例，从而简化为支付方和收讫方设置支付指南的复杂性。另一方面，开发者也可使用 FPS 的综合 API 集[94]来支持某些应用更精细的支付要求，开发者可以使用这些更基础的 API 来创建更独特的支付指南。

在正式上线前，开发者可利用 FPS 的沙箱[95]来开发并测试应用。

9.1.3 特点

Amazon FPS 具有如下特点：

- 向 Amazon 客户收费的最简单途径：基于 FPS，成千上万的现有 Amazon.com 客户可以使用同样的账号和支付方法在开发者自己的网站上完成支付操作。
- 灵活：每笔 FPS 交易都由一个支付方、收讫方、调用者（用 API 调用 Amazon FPS 的一方），当开发者在自己网站使用 FPS 来收取交易资金时，收讫方和调用者合并一起，都是开发者本身。交易中的双方都可能想给交易添加条件或规则（即支付指南），支付指南的例子包括：
  - 交易数量：指定某笔支付的最小、最大、范围或特定数量。
  - 交易日期：配置支付交易在特定时间发生（例如，某天、每周、每月或某个日期范围）。
  - 消费限制：设置每日、每周或每月的交易数目或消费总额限制，来控制自己应用的开销。
  - 接收方列表：指定授权谁可使用或接收资金。
  - 支付方法：指定支付方法（信用卡、银行卡、Amazon 支付余额划账或其组合）。
  - 费用：控制哪一方支付 Amazon FPS 的交易费用。
- 降低成本：FPS 利用 Amazon 可信的欺诈检测功能、拒付控制、风险管理流程等手段来降低呆坏账。对在开发者网站上消费的用户而言，他们也可放心购买，因为他们处于和在 Amazon.com 上购物同样的保护之下[96]。

9.1.4 定价

用户可免费使用 Amazon FPS 沙箱，FPS 的收费与每笔交易的支付方式、交易额有关：

- 使用 Amazon 支付余额：每笔交易收取交易额的 1.5%，另加 0.01 美元；
- 使用银行卡：每笔交易收取交易额的 2.0%，另加 0.05 美元；
- 使用信用卡：10 美元以上的，每笔交易收取交易额的 2.9%，另加 0.3 美元；不足 10 美元的，每笔交易收取交易额的 5%，另加 0.05 美元。

对每笔不足 0.05 美元的微型交易，有多种收费模式[97]，另也有若干折扣形式[98]。
以上为美国用户的收费标准，非美国用户，每笔交易另收交易额的 1%作为费用。

9.1.5 产品细节

1) 支付指南：FPS 的组建模块

交易各方可使用 Amazon FPS 提供的 Gatekeeper（GK）语言来创建交易指南，一旦开发者定义了这些交易指南，可使用 *InstallPaymentInstruction* API 来安装这些指南，并收到一个 token 作为唯一的安全机制来处理支付指南。

调用方可使用带有支付方、收讫方及调用方 token 的 *Pay* API 来执行支付交易，也可调用 *Reserve* API 来预授权支付方信用卡上的资金，并在商品或服务成功交付后，调用 *Settle* API 从支付方信用卡上扣除该笔费用。

支付指南借助和独立的第三方支付系统，FPS 提供了无与伦比的灵活性来构建创新的支

付应用，开发者可使用 Quick Starts 包来支持应用，或使用 API 提供更精细的支付操作。

2) 使用 Amazon FPS 沙箱

FPS 提供了全功能的沙箱环境，用于在不发生真实交易的前提下，开发和测试应用或服务，开发者可使用 AWS 账户访问沙箱。FPS 沙箱提供一个 Web Service API endpoint 来测试 API 的调用，开发者可以模拟重要的支付出错场景。开发者还可访问沙箱网站来创建并管理测试账号。

Amazon 的结账（Checkout）解决方案[99]，可以方便地集成到用户网站以提供安全可靠的点击结账的功能。而 Amazon 简单支付方案[100]提供了一组 HTML 按钮，用户可通过拷贝粘贴的方式快速集成到自己的网站上，使网站客户能重用在 Amazon 已有的支付信息。相较以上两种 Amazon 支付产品，Amazon FPS 有所不同，FPS 提供了更高层次的可定制的支付解决方案，适用于各种商业应用。

9.2 Amazon DevPay

9.2.1 简介

Amazon DevPay[101]是一个在线的计费与账号管理服务，使企业和开发者能更方便地售卖基于 AWS 的应用。对于传统的在线订购服务或应用而言，创建并管理订单流水线或者计费系统是很大的一个挑战，但采用 DevPay 后，用户就不会再有这样的痛苦了。企业或开发者可以使用 DevPay 快速实现诸多功能，如客户注册、自动计量客户的 AWS 服务使用量并据此生成 Amazon 账单、以及收讫支付等。Amazon DevPay 提供了 Web Service 接口，可按多种付费方式以及服务使用情况，为开发者所开发的应用计价，并使用 Amazon Payments 来处理来自应用客户的支付，成千上万的 Amazon 客户在使用了开发者开发的应用后，可以使用已有的 Amazon 账号来支付。

9.2.2 功能

如果开发者在 EC2 上有一些 Amazon EC2 虚拟机镜像文件（AMI）或一些使用 S3 的应用相对外销售，则可通过 Amazon DevPay 来管理终端客户对这些文件或应用的订购与计费。Amazon DevPay 的使用非常简单，步骤如下：

- 使用 S3 或 AMI 来存放所开发的软件应用。
- 使用 DevPay 的 Web Service 接口来注册上述应用或 AMI，提供产品描述和价格。
- 将 DevPay 的购买链接加入到你的网站中，使客户能购买你的产品。
- 集成 DevPay 的认证机制，保证只有认证的客户可以访问你的应用。
- 在 Amazon DevPay Activity 页面[102]监控业务运行情况。

9.2.3 特点

Amazon DevPay 具有如下特点：

- 易于使用的在线计费与账户管理系统
  开发者可使用一个 Web Service 接口来注册 DevPay 服务，添加自己的应用描述信息以及价格。DevPay 会随之提供一个链接地址，开发者可把该链接地址放置在自己网站的任何位置，以引导消费者完成 DevPay 管理下的购买流程。DevPay 负责自动跟踪消费者对 AWS 资源的使用情况，生成账单，收取费用并将属于开发者收入的一部分直接打到开发者专有的 Amazon 支付账户（可在注册 Amazon DevPay 服务时创建）中。
- 灵活的产品定价选项：开发者可以自由决定如何为自己的应用定价，即可按一次性付费，也可每月定期付费。DevPay 会自动计量消费者对 AWS 的使用情况，这样开发者就可以据此来决定要收取消费者多少使用费用。开发者也可随时调整价格。
- 可信：开发者使用 DevPay 来支撑应用的另一个好处在于，成千上万的消费者使用其在 Amazon 的账户信息即可完成对应用的付费操作，这样，既使消费者更方便地消费应用，也使支付更加安全可信。

9.2.4 定价

开发者的收入来自于消费者的增值，即除了使用 AWS 服务（如 S3 等）之外（这部分费用由消费者支付，开发者无需负担，并归 Amazon 所有），消费者因使用开发者的应用而另需支付的费用。开发者从每个消费者获得的增值收入的 3%为使用 DevPay 的费用，此外，消费者使用每个应用而生成的每笔账单另收取 0.3 美元。

9.2.5 产品细节

1) 注册应用与定价

点击 DevPay 页面（https://aws.amazon.com/devpay/）的“Sign Up For Amazon DevPay”按钮，将自己的应用注册到 DevPay。注册中，可以描述应用信息并设定价格，定价方式可以包括下列任意一种或所有收费类型：

- 一次性付费：在消费者注册应用的当天就一次性扣取相应的注册费用。
- 每月定期付费：每月支付固定的应用使用费用，每月初会将账单发给订购客户。注册应用时会按比例收取首月的费用。
- 根据 AWS 的使用情况计费：费用与 AWS 的使用情况相绑定。既可按 Amazon 的 AWS 计费标准来收费，例如，每月使用的每 GB 存储、每 GB 的数据传输量等，也可自己设定收费标准，例如，使用 S3 时每传输 1 GB 数据收取 0.5 美元等，尽管实际上 S3 中每传输 1 GB 数据 Amazon 只收取 0.1 美元。DevPay 按开发者定义的价格标准收取消费者的费用，并把消费者使用 AWS 服务而应支付的费用账单发送给开发者。

2) 开发者应用与 DevPay 的集成

要将自己的 S3 应用与 DevPay 集成起来，开发者只需在已有的 S3 API 调用上增加两个参数：“product token”和“user token”。Product token 是应用的标识符，在把应用注册到 DevPay 时获得此值。User token 是要使用应用的消费者的标识符，DevPay 也按此 token 计费，开发者可调用 *ActivateHostProduct* 或者 *ActivateDesktopProduct* API 来获取 user token。

要将自己的 AMI 与 DevPay 集成起来，需要把 AMI 和 product code 联系起来。Product code

可看作应用的句柄，在把应用注册到 DevPay 时获得此值。使用 EC2 命令行工具或 *ModifyImageAttribute* API 来关联 product code 和 AMI。之后，消费者就可以通过 DevPay 购买 AMI，并调用 EC2 API 启动 AMI，整个调用和启动过程与使用其它 AMI 一致。

应用或 AMI 与 DevPay 集成，DevPay 会确保只有账户正常的消费者才可以使用应用，DevPay 还会自动计量消费者的使用情况并根据开发者设定的价格标准来计费。

更多与 DevPay 集成的信息请参考 DevPay 使用指南[103]。

3) 销售应用

开发者完成应用注册和 DevPay 集成步骤后，要将 DevPay 为产品购买流程提供的一个链接地址嵌入到自己的网站上。点击该链接，会将消费者引导到 Amazon DevPay 页面，消费者使用自己的 Amazon.com 凭证登录并选择一张存储在自己 Amazon.com 账户中的信用卡进行消费，购买完成后，消费者再重新被引导回开发者自己的网站。这与我们所熟知的银行卡、信用卡、支付宝等在线支付流程和方式是类似的。

4) 支付 AWS 服务的使用费用

DevPay 生成消费者的账单后，会统一向开发者收取使用 AWS 服务的费用，如果某个消费者没有支付其应该支付的使用 AWS 服务的费用，Amazon 不会将这笔费用转嫁到开发者头上。但如果按照定价，消费者要支付给开发者的费用甚至低于使用 AWS 服务的费用时，开发者就必须自掏腰包弥补这个差额，即使消费者没有支付亦是如此。所以，开发者要基于 AWS 的服务和 DevPay 的使用价格，留出自己的盈利空间（增值收入）。

4) 业务监控与价格调整

开发者使用 DevPay 售卖自己的应用后，可在 Amazon DevPay Activity 页面[102]监控应用的收入情况，并使用 DevPay 的 Web Service 接口来调整价格及生效时间。DevPay 会向应用的消费者发送 email 通知价格的改变，开发者可以定制 email 内容。

5) Amazon DevPay 价格

可参见第 9.2.4 节。Amazon DevPay 每个消费者增值收入的 3%，增值收入指的是从消费者收取的总收入减去消费者使用 AWS 服务的费用。只有当总收入大于 AWS 开支时，DevPay 才会收取 3%的费用。此外，每个应用的每笔账单 DevPay 都会收取 0.3 美元费用。下面是一个简单的例子：

- 来自某个消费者的收入为 150 美元
- 该消费者消费的 AWS 为 70 美元
- 所以该消费者的增值收入 150 - 70 = 80 美元
- DevPay 收取的费用为 80 * 3% + 0.3 = 2.7 美元
- 所以，来自该消费者的净收入为 150 – 70 – 2.7 = 77.3 美元

第十章 存储类（Storage）

10.1 Amazon Simple Storage Service (S3)

10.1.1 简介

Amazon S3[5]设计为 Internet 的存储设备，使 Web 扩展变得更为容易。S3 提供了一个简单的 Web Service 接口，用户可用该接口来在任意时间从 Web 的任何地点存储并获取任意数量的数据。S3 所基于的底层架构和 Amazon 网站所使用的全球网络是一样的，任何开发者都可以访问这样一个高扩展、可靠、安全、快速和经济的基础架构。

10.1.2 特点

Amazon S3 具有如下特点：

- 可以读、写、删除包含 1 byte 到 5 TB 数据的对象，用户可存储的对象数量没有限制。
- 每个对象都存储在一个 bucket 中，并通过一个唯一的由开发者指定的 key 来读取。
- 一个 bucket 存储在某个 Region 中，用户可以选择存储的 Region，以优化延迟、减少开支、满足管理需求等。S3 目前在下列 Region 中可用：美国标准、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）、欧盟（爱尔兰）、亚太（新加坡）、亚太（东京）、南美洲（圣保罗）以及 GovCloud（美国）[104]（GovCloud 是 Amazon 专为美国政府部门机构及合同商提供的云计算服务环境，本文将在 AWS 解决方案章节详细介绍 GovCloud）。US 标准 Region 自动通过网络拓扑将请求路由到弗吉尼亚州北部或太平洋西北部的 S3 设备上。
- 存储在某个 Region 中的对象不会离开该 Region，除非用户主动将其迁移出去。
- S3 提供了认证机制保证未经认证的请求不能访问数据。数据可设为私有或公有类型，并可为每个数据使用者赋予不同的访问权限。此外，S3 还提供了数据安全上传/下载及静态数据加密选项。
- S3 采用标准的 REST 和 SOAP 接口访问。
- 默认下载协议是 HTTP，亦提供了 BitTorrent 协议接口以满足高扩展分发的需求。
- 提供定期删除和大容量删除的选项。对定期删除，可以定义规则，在预设置的时间到达后，删除一组对象，一条请求最多可以一次性删除 1000 个对象。
- S3 的可靠性由 Amazon S3 SLA[105]提供保证。

10.1.3 定价

S3 提供了免费使用的配额，超过配额部分的，费用与所在 Region、存储空间大小、请求数量、数据传输量等有关[5]。GovCloud 上的 S3 定价稍有不同[104]。

10.1.4 常用场景

Amazon S3 可用于大量应用场景，例如：

- 内容存储与分发
  S3 可为 Web 应用、媒体文件等提供高持久性、高可用性的存储支持，随着应用数据的增加，可以随时动态扩展存储空间。用户可直接从 S3 上分发自己的内容，也可将 S3 作为数据源，并将内容发布到 Amazon CloudFront 的节点服务器上。
  如果要分享的数据内容易于重新生成或者在其它地方已有备份，S3 提供了低成本的冗余精简存储（Reduced Redundancy Storage，RRS）方案。例如，用户公司内部有一份媒体内容的存储，但为了让公司客户、渠道合作商或员工能更方便地访问这些内容，可以采用 RRS 来存储并分享这些数据。
- 存储数据分析的结果
  Amazon S3 非常适合存储某些应用场景的原始数据内容，例如制药分析数据、用于计算和定价的金融数据，或用于处理的相片等。这些数据可被直接发送到 EC2 用于计算或其它大规模的分析之用，EC2 和 S3 之间的数据传输完全是免费的。这些可重复生成的处理结果既可采用 S3 标准方式存储，也可使用 RRS 特性存储。
- 备份、归档与灾难恢复
  S3 为用户的关键数据提供了高持久性、高扩展性以及安全的备份和归档方案，甚至还可使用 S3 的版本控制功能为存储的数据提供更多的保护。如果数据量非常庞大，可以使用 AWS 导入/导出功能[106]将大量数据从物理存储设备导入 S3 或从 S3 导出到物理存储设备。这非常适合周期性的备份以及灾难恢复时的数据快速获取。

10.1.5 产品细节

1) 设计需求

S3 的目标是简单而鲁棒：廉价而安全地存储任意数量的数据，并且数据随时可用，为此 S3 实现了以下设计需求：

- 安全：用户可以完全控制谁可以访问他们的数据，数据在传输和存储时也必须安全。
- 可靠：存储的数据具有 99.999999999%的可靠性以及 99.99%的可用性，不存在单点故障以及宕机时间。
- 可扩展：通过添加新的节点到系统中，S3 可从存储空间、请求速率及用户数几方面支持任意多个 Web 扩展的应用。
- 快速：S3 足够快速，能支持高性能需求的应用，S3 服务器端的延迟远远小于 Internet 的网络延迟，任何性能瓶颈都可以通过往系统中添加节点来解决。
- 廉价：S3 构建在廉价的硬件设备上，任何硬件的故障都不会影响整个系统的性能。
- 易于使用

S3 设计中有一个强制性要求，即单个的 S3 分布式系统必须既支持 Amazon 内部的应用，也要支持外部开发者的任何应用。这意味着 S3 必须足够快速可靠来运行 Amazon.com 网站，同时又要足够弹性以满足任意开发者用来存储任意数据。

2) 使用入门

S3 最基本的使用步骤如下，更多细节请参考 S3 使用指南[107]：

- 创建一个 bucket 来存储数据，可选择 bucket 和 object 所在的 Region。

- 上传 object 到 bucket 中，数据即可持久存储，并受 S3 SLA 保证。Bucket 创建和 object 上传都是安全的。
- 若需要，可使用 AWS 管理控制台、AWS .NET SDK[108]、AWS Java SDK[109]设置访问控制策略。

3) 数据保护

S3 默认对存储其上的数据提供安全保护，只有 bucket 及 object 的所有者才能访问他们创建的 S3 资源。S3 支持多种访问控制机制，如传输数据加密和盘上数据加密。对于必须遵从诸如 PCI、HIPAA 等管理标准的用户，S3 的数据保护特性可作为整体策略的一部分。S3 的数据安全与可靠性特性描述如下。

- 数据安全细节

  S3 提供了四种不同的访问控制机制，用于控制谁在何时何地以何种方式访问数据：
  - IAM（Identity and Access Management）策略：IAM 允许在一个 AWS 账号下创建并管理多个用户，适合有若干员工的组织使用。基于 IAM 策略，可以对 IAM 用户访问 S3 bucket 或 object 给予更精细的控制。
  - ACL（Access Control List）：可为单个 object 选择性地添加某种权限。
  - Bucket 策略：可用来添加或剥夺某个 bucket 中部分或全部 object 的权限。
  - 查询字符串认证：基于查询字符串认证，能通过 URL 来共享 S3 object，只要该 URL 在预定义的过期时间内有效。

  用户可基于 HTTPS 协议，通过 SSL 加密 endpoint 来安全地上传数据到 S3 或从 S3 下载数据。S3 也为盘上数据提供了多种加密选项：
  - 如果希望自己管理加密密钥，可在上传数据到 S3 之前使用客户端加密库（如 Amazon S3 加密客户端[110]）来加密数据。
  - 如果希望 S3 来管理加密密钥，可使用 S3 服务器端加密（Server Side Encryption，SSE）。使用 S3 SSE 时，只要在写 object 时增加一个额外的请求头，即可实现上传时加密数据，S3 会在收到数据时自动执行解密操作。

  S3 还支持日志功能，用户可配置 S3 bucket 在收到访问请求时创建日志记录，记录对 S3 bucket 或 object 的所有访问请求，以用于审计目的。

  有关 S3 的更多安全特性信息，可参考 Amazon S3 开发者指南[111]。如果要了解包括 S3 在内的所有 AWS 的安全特性，请参考 AWS 安全处理概览文档[12]。
- 数据持久性与可靠性

  S3 作为高持久性存储基础架构，设计为关键及主存储设备之用。S3 object 会在一个 S3 Region 的多处场所的多台设备上冗余存储三份拷贝。为了确保数据持久性，S3 的 PUT 和 COPY 操作在返回 SUCCESS 之前，会同步地在多处场所存储数据。存储完成后，S3 负责维护 object 的持久性，一旦监测到冗余备份有丢失，会立刻恢复应有的备份数。S3 还会定期用校验和（checksum）来校验存储数据的一致性。一旦发现有误，就用冗余数据予以恢复。此外，S3 会对所有网络流量计算校验和，来保证读写数据时能发现损坏的数据包。

  S3 存储方式有标准存储和 RRS 两种，标准存储具有如下特点：
  - 受 Amazon S3 SLA 支持。

- 在给定年限内，为 object 提供 99.999999999%的可靠性以及 99.99%的可用性。
- 能承受存储在两个场所的同一份数据同时丢失。
- 提供版本控制，可用来保存、获取并恢复存储在 bucket 中的每个 object 的每一个版本，使用户能轻松地从意外操作或应用故障中恢复。默认情况下，S3 请求会读取最新写入的版本，可在请求中指定版本来读取旧版本的 object。

冗余精简存储（RRS）作为 S3 的一个存储选项，允许用户在存储非关键及可重复生成数据时（如数据在其它地方还有持久化的存储、缩略数据、转码生成的媒体文件及其它很容易重复生成的数据），采用低于 S3 标准存储的冗余级别，以降低存储开支。RRS 在多个场所的多个设备中存储 ojbect，可靠性是典型的本地磁盘存储的 400 倍，但 RRS 冗余备份数为两份。RRS 存储具有如下特点：

- 受 Amazon S3 SLA 支持。
- 在给定年限内，为 object 提供 99.99%的可靠性以及 99.99%的可用性，这意味着 object 每年丢失的平均概率为 0.01%。
- 能承受在单个场所中存储的数据丢失。

4) 大数据传输

使用 AWS 导入导出功能可以加快大数据量进出 AWS，AWS 使用 Amazon 的高速内部网络来传输用户数据。对大数据集，AWS 导入导出功能通常要快于 Interent 传输，故适合用户数据迁移至云端、向客户分发数据、向 AWS 发送备份以及灾难恢复等操作。

10.2 Amazon Glacier

10.2.1 简介

Amazon Glacier[112]为归档和备份数据提供了极低费用的安全持久的存储服务。为了降低费用，Glacier 针对不经常访问的数据做了专门的优化，这类数据通常不会实时获取，而要等待几个小时。Glacier 的数据存储价格为每月每 GB 只要 0.01 美元。

公司一般在数据归档上超额开支。首先，它们被迫为其归档方案支付昂贵的费用（这还不包括运营成本，如电力、设施、人员及维护费用等）。其次，为了提供足够的数据冗余并应对未知的数据增长，公司常常要部署超过当前需要的硬件设施，从而降低了利用率，亦浪费了资金。而借助 Glacier，用户可以按需付费，并按需向上或向下扩展，费用极其低廉。

上传到 Glacier 的数据称为归档文件，既可代表一个单独的文件，也可将若干文件作为一个归档文件打包上传。从 Glacier 获取归档文件，需要提交一个 job，通常要 3-5 小时完成。Glacier 将归档文件放在 vault 中，可使用 Amazon IAM 服务来控制对 vault 的访问。

Amazon Glacier 的简单使用方法如下，更多信息请参考 Amazon Glacier 入门视频[113]或 Amazon Glacier 开发者指南[114]。

- 使用 AWS 管理控制台或 Amazon Glacier API 来创建 vault，vault 用来存储上传到 Glacier 的归档文件。
- 使用 Amazon Glacier API 上传或读取归档文件。
- 使用 Amazon Glacier API 监控 Glacier job 的运行状态，也可配置 vault，当 job 结束时，通过 Amazon SNS 发送一条通知。

- 按需付费，费用基于每月存储和传输的数据量。

10.2.2 特点

Amazon Glacier 具有如下特点：

- 费用低：按数据存储和传输量付费，没有任何其它额外的费用，最低可到每月每 GB 存储只要 0.01 美元。
- 安全：Glacier 支持数据通过 SSL 传输，并且自动使用高级加密标准（Advanced Encryption Standard，AES）256 为盘上存储的数据加密，AES 256 是一个安全的对称密钥加密标准，使用 256 位的加密密钥。用户也可使用 AWS IAM 来控制对数据的访问。基于 IAM，一个 AWS 账号下可以创建并管理多个用户，并基于资源制定访问策略。
- 持久性：Glacier 设计目标为归档文件平均每年的可靠性为 99.999999999%。Glacier 会在多个场所中冗余存储数据，且每个场所都会在多台设备上冗余存储。在上传归档文件返回 SUCCESS 之前，Glacier 会同步在多个场所中存储数据。Glacier 还会执行定期、系统化的数据一致性校验，且可自愈性地解决发现的问题。
- 灵活：用户在 Glacier 存储的数据没有大小限制，无论是 PB 还是 GB 级，Glacier 都可以实现向上或向下地自动扩展。而且，用户可以根据实际情况选择将数据存储在哪个 Glacier Region。
- 简单：用户可将任意大小的数据在 Glacier 上存储任意长的时间，而无需考虑管理、运营和硬件故障带来的各种问题和费用开支。
- 与其它 AWS 集成：可使用 AWS 导入导出功能加快 AWS 和便携式存储设备之间的大量数据交互。S3 未来数月内还会计划推出一个新特性，可使用数据生命周期策略，让数据可以在 S3 和 Glacier 之间无缝迁移。

10.2.3 定价

Glacier 的费用来自以下几部分：

- 存储费用：最低可至每月每 GB 只收取 0.01 美元，视 Region 不同而费用稍有差别。
- 请求费用：每 1000 个上传或读取请求收取 0.06 美元，其它请求免费。
- 数据传输费用：数据传入 AWS Region 免费，数据传出 AWS Region 的每月首个 GB 免费，之后每月的 10 TB 收取每 GB 0.201 美元，之后的分段收费逐步减少，具体价格请参考 Glacier 主页[112]。在同一 Region 的 EC2 和 Glacier 之间传输数据免费。

10.2.4 常用场景

Amazon Glacier 有广泛的应用场景，例如：

- 企业信息归档
  由于商业和监管的需要，以及产出数据的增长，企业不得不归档越来越多的数据，例如 email、合法记录、金融及商业文档等。这些数据通常要保留数年甚至数十年，但并不经常访问。Amazon Glacier 使企业能将大量数据以极低成本存储，并不再承担任何数据存储的附加成本。

- 媒体资产归档
  媒体公司的核心资产就是它们的内容，包括书籍、电影、音乐、图片、新闻电影胶片以及电视节目等。随着新的制作以及大量使用诸如高分辨率电视、社交媒体和3D 视频之类的新技术，这些资产的数目和大小还在持续增长，可达到几十甚至几百 PB，安全地存储这些资产就变得极其的重要。此外，数据可访问性也至关重要，例如，因为某个突发事件，某个已归档的新闻电影胶片可能突然之间变得非常有价值，此时就必须能够在合理的时间内读取到已归档的数据。访问 Glacier 上存储的媒体文件非常简单，只要调用服务的 API 即可。
- 科研数据归档
  科研机构，如医药和生物公司、大学和研究所等，都有归档大量数据的需求。例如，在新药开发中，会产生可观的数据，这些数据都要保留下来，以便研究人员可以验证实验药品的测试结果。通常，这些数据存储在磁带设备上，并在多处数据中心备份存储。借助 Glacier，可以满足所有这些需求。Glacier 会在多个场所中冗余存储数据，且每个场所都会在多台设备上冗余存储。Glacier 还会执行定期、系统化的数据一致性校验，且使用冗余数据来自愈性地解决发现的问题。
- 数码收藏
  数码收藏主要常见于诸如图书馆、史学社团、非营利性组织及政府部门，它们的工作是收藏那些有价值的年代久远的数码内容，如网站、软件源代码、视频游戏、用户生成内容及其它不再容易得到的数字产品。这些归档文件刚开始时可能不大，但可能随时间而出现 PB 级的增长。Glacier 的自动扩展功能完全可以满足这种需求。
- 磁带机的替代品
  Amazon Glacier 可以替代定制或外包的磁带库。尽管磁带存储具有廉价、可扩展的特点，但其资源总有枯竭受限的时候，因为磁带库需要专人维护，并且占用数据中心的宝贵空间，而且磁带本身需要小心保存和管理，包括定期的将数据从旧磁带拷到新磁带上。此外，磁带库的设备扩容也是一件很难做到精确完美的事情，常常伴随使用率降低、投资浪费的问题。采用 Glacier 代替磁带库，可以不再承担这些运行、管理及扩容成本与风险，用户随时具备高可扩展能力，存储任意数量的数据。借助 AWS 导入导出功能，可以经济快速地将磁带库的内容迁移到 Glacier 上。

10.3 Amazon Elastic Block Store (EBS)

10.3.1 简介

Amazon 弹性块存储（Elastic Block Store，EBS）[9]为 EC2 实例提供块级别的存储卷服务。Amazon EBS 卷是网络连接的持久化独立存储设备，不受 EC2 实例的生命周期影响。EBS 尤其适合需要数据库、文件系统、或访问原始块存储设备的应用。

Amazon EBS 卷创建在某个 AZ 中，大小可为 1 GB 到 1 TB，可连接到同一 AZ 中的任意 EC2 实例上，连接成功后的 EBS 卷可看作一个 mount 上的设备，和任何其它硬盘或块设备没有什么区别。EC2 实例可以像操作本地磁盘一样使用该卷，格式化卷为某种文件系统或者直接安装应用。

EBS 卷某一时刻只能连接到一个实例，但一个实例可同时连接多个卷。这样，用户可以往一个实例上连接多个 EBS 卷，并将数据分散存储在这些卷上，以提供 I/O 和带宽性能，

这尤其适合数据库类型的应用，因为在数据集上经常有大量随机的读写操作。如果某个实例出现故障或解除了到 EBS 卷的连接，该卷就可以连接到该 AZ 中的任意其它实例上。

EBS 卷也可作为 EC2 实例的根分区，这样 EC2 实例的根分区大小最大可以到 1 TB，并可在销毁实例时保存下根分区数据，以便和其它 AMI 绑定。此外，当为保存状态而停止并重启根分区为 EBS 卷的实例时，启动时间非常快。

10.3.2 特点

Amazon EBS 具有如下特点：

- EBS 卷大小为 1 GB 到 1 TB，可 mount 为 EC2 实例的设备，多个 EBS 卷可 mount 到同一实例。
- 通过选择 Provisioned IOPS 卷，EBS 可部署特定级别的 I/O 性能，能使每个 EC2 实例扩展到数千 IOPS。
- EBS 存储卷看似一个原始的、未格式化的块设备，由用户提供设备名及块设备接口。用户可像使用硬盘一样，在 EBS 卷上创建一个文件系统，或者任何块设备支持的操作。
- EBS 卷可连接到同一 AZ 中 EC2 实例上。
- 每个 EBS 存储卷都会自动在同一 AZ 中备份，避免单点硬件故障。
- EBS 支持创建卷的快照，并存储在 S3 中。这些快照可作为起点，初始化任意多个新的 EBS 卷。
- AWS 支持用户基于 AWS 上的公众数据集[115]来创建新的 EBS 卷。
- Amazon CloudWatch 提供了 EBS 卷的性能计量参数，包括带宽、吞吐率、延迟、队列长度等。可通过 AWS CloudWatch API 或 AWS 管理控制台来访问这些计量值[10]。

10.3.3 定价

Amazon EBS 的费用由三个方面组成：

- Amazon EBS 标准卷
  - 每月每 GB 收取 0.12 美元
  - 每 1 百万 I/O 请求收取 0.12 美元
- Amazon EBS Provisioned IOPS 卷
  - 每月每 GB 收取 0.15 美元
  - 每月每个 Provisioned IOPS 收取 0.12 美元
- 在 S3 上存储的 Amazon EBS 快照
  - 每月每 GB 收取 0.13 美元

10.3.4 产品细节

1) Amazon EBS 卷的性能

Amazon EBS 提供了两种类型的卷：标准卷和 Provisioned IOPS 卷，它们在性能特征和价格上有所区别。

- 标准卷适合为中等或突发 I/O 需求的应用提供存储服务。标准卷支持的平均 IOPS 大致为 100，突发时最好可到数百 IOPS。标准卷非常适合做根分区卷，对突发 I/O 的支持能力使实例的启动时间非常迅速。
- Provisioned IOPS 卷适合可预测、高性能的 I/O 密集型应用，如数据库等。在创建 Provisioned IOPS 卷时，可以指定 IOPS 速率，EBS 会在该卷的存续期内提供相应的速率。目前 EBS 支持每个 Provisioned IOPS 卷最高 1000 IOPS，未来很快还会提高此限制数。通过连接多个 IOPS 卷到同一 EC2 实例上，应用可支持数千 IOPS。为了使 EC2 实例能够充分利用 IOPS 卷的 I/O 能力，可以启动“EBS 优化”类型的 EC2 实例[18]，EBS 优化实例可在 EC2 和 EBS 之间提供 500 Mbps 至 1000 Mbps 的吞吐率。

实际的 EBS 性能与应用相关，建议用户基于 EC2 实例类型和 EBS 类型，制作自己应用的 benchmark。有关 EBS 性能的更多信息，请参考 EC2 用户指南的 EBS 章节[116]。

2) Amazon EBS 卷的可靠性

EBS 卷设计为高可用和高可靠的存储设备。EBS 卷的数据在 AZ 内的多台服务器上备份存储。EBS 卷的可靠性依赖于卷的大小以及上次快照后变更数据的百分比，例如，一个 20 GB 的卷，或者最近一次 EBS 快照后没有什么数据改动的情况下，预期的每年故障率（Annual Failure Rate，AFR）在 0.1% - 0.5%之间，此处的“故障”表示完全丢失卷，这个值已经 10 倍强于普通硬盘 4%左右的 AFR 了。

除了 EBS 在 AZ 内冗余备份存储，EBS 还提供了创建卷的快照的功能，快照存储在 S3 上，并自动拷贝到多个 AZ 中，因此经常为卷打快照是提高数据长期可靠性的一个方便经济的途径。万一 EBS 卷出现故障，用户可以从最近的快照重新创建卷。

3) Amazon EBS 快照

EBS 快照是增量备份，即只保存那些有在上次快照后有所更改的块。虽然 EBS 快照是增量保存的，但当删除一个快照时，只会删除那些不会被任何其它快照使用的数据。因此，无论删除了之前的哪个快照，所有现有的活动快照都仍包含恢复卷所需的全部信息。此外，恢复卷所需的时间对所有快照而言都是一样的。

快照还可用来初始化 EBS 卷、扩展已有卷的大小或在 AZ 之间移动卷。创建 EBS 卷时，有一个选项，允许基于现有的 S3 上的快照来生成卷，这样，新生成的卷是原始卷完全相同的一份拷贝。如果使用快照来改变卷大小，必须确认文件系统或应用支持改变设备的大小。

从 S3 上快照创建一个新的卷是一个懒加载过程，即并不需要等到所有数据都从 S3 迁移到新的 EBS 卷上，才告知卷创建完成，EBS 卷会在后台执行数据加载操作。如果 EC2 实例访问的数据片段还未被加载到 EBS 卷上，EBS 卷会立刻从 S3 下载所请求的数据，然后继续在后台加载其它的数据。

借助 EBS 的快照特性，用户可以公开自己的数据，允许哪些授权 AWS 用户共享访问，具有访问权限的用户即可基于这些快照来创建自己的 EBS 卷。

10.4 AWS Import/Export

10.4.1 简介

AWS 导入导出功能[106]加快了大批量数据在 AWS 和便携式存储设备之间的传输。AWS 使用 Amazon 的高速内部网络而非 Internet 来传输数据，一般来说，AWS 导入导出功能要快于 Internet 的传输速率，也更为经济。

AWS 导入导出功能支持以下功能：

- 数据导入导出 S3 bucket，支持的 Region 包括：美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）、欧盟（爱尔兰）、亚太（新加坡）。
- 数据导入到 EBS 快照，支持的 Region 包括：美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）。
- 数据导入到 Glacier，支持的 Region 包括：美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）、欧盟（爱尔兰）。

点击 AWS 导入导出主页的“Sign Up for AWS Import/Export”按钮即可开始使用该服务。使用步骤也很简单，包括：

1) 准备一个便携式存储设备，支持的设备请参见第 10.4.4 节。
2) 向 AWS 提交一个创建作业的请求，请求中包含的信息有 S3 bucket、EBS 或 Glacier 所在的 Region、AWS Access Key ID 及自己的邮政地址。提交成功后，可收到返回信息，包括作业的唯一标识符、认证便携式存储设备的数字签名，以及存储设备要寄送的 AWS 地址。
3) 对第 1)步中准备的便携式存储设备进行安全识别与认证操作。对 S3，将第 2)步中获得的签名文件拷贝到存储设备的根目录下；对 EBS 或 Glacier，将签名条形码粘在存储设备的外表。
4) 将存储设备与连接插头、电源一起邮寄给 AWS。
5) AWS 收到包裹后，会安全送到相应的 AWS 数据中心，并连接到 AWS 导入导出工作站。数据导入导出完成后，存储设备会按第 2)步请求中设定的地址寄回。

10.4.2 定价

AWS 导入导出的费用与处理的便携式存储设备个数以及耗费的处理小时数有关。

- 每个便携式存储设备收取 80 美元处理费用。
- 数据导入导出时，每小时收费 2.49 美元，不足一小时按一小时算。
- 如果回寄地址在美国国内，AWS 将支付回寄费用。如果要求快递或回寄地址在美国以外，将根据回寄地址和重量计费。
- 在 AWS 导入导出服务与 S3、Glacier、EBS 之间没有数据传输费用。

10.4.3 常用场景

AWS 导入导出服务的常用场景包括：

- 数据迁移：如果用户有大量数据需要首次上传到 AWS 云端，可以考虑使用 AWS 的导入导出服务，其较通过 Internet 来上传要快得多。

- 内容分发：将大量数据通过便携式存储设备发送给客户。
- 数据直接交换：如果定期从商业伙伴处收到包含数据内容的便携式存储设备，可以直接将设备送到 AWS，以导入到 S3 或 EBS 中。
- 外部备份：将全部或增量备份文件发送到 S3 及 Glacier，以获取可靠冗余的外部备份存储。
- 灾难恢复：在故障或灾难恢复中，如果需要立刻从 S3 或 Glacier 中获取大批量的备份数据，可以使用 AWS 导入导出服务将数据传输到便携式存储设备上，并寄回。

10.4.4 产品细节

1) 何时使用 AWS 导入导出服务

当有大批量数据要上传至 AWS 或从 AWS 下载，而 Internet 带宽有限，使得数据在 Internet 的传输时间远远大于邮寄便携式存储设备到 AWS 的时间时，比如 Internet 传输数据超过一周或更多时，建议考虑使用 AWS 导入导出服务。

表 10-1 列出了在一般 Internet 连接状况下：(1) 从 Internet 传输 1 TB 数据到 AWS 要花费的时间；(2) 通过 Interent 要耗费一周时间才能传输完的数据量，此时建议考虑 AWS 导入导出服务。更多数据加载开支细节请参考 AWS 导入导出计算器[117]。

表 10-1 AWS 导入导出服务选择时机建议

Interent 连接	**在 80%网络带宽利用率的前提下，传输 1 TB 数据的理论最小天数**	**何时考虑使用 AWS 导入导出服务**
T1 (1.544Mbps)	82 天	100 GB 或更多
10Mbps	13 天	600 GB 或更多
T3 (44.736Mbps)	3 天	2 TB 或更多
100Mbps	1 – 2 天	5 TB 或更多
1000Mbps	不足一天	60 TB 或更多

2) 便携式存储设备

为了能保证 AWS 导入导出服务可用，必须使用兼容的便携式存储设备，按照所在 Region 的不同，美国标准、美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（加利福尼亚州北部）Region 的需求如表 10-2 所示，欧盟（爱尔兰）Region 的需求如表 10-3 所示，亚太（新加坡）Region 的需求如表 10-4 所示：

表 10-2 美国本土 Region 的便携式存储设备要求

项目	**需求**
设备的外部电源要求	美式壁插座插头（其它形式插头请附寄转换器） 120 伏 / 60 赫兹

项目	需求
	最大电源功率 2,000 瓦
接口类型	eSATA USB 2.0 或 3.0 （邮寄包裹需包含 USB flash 驱动） 2.5（6.4cm）或 3.5 英寸（8.9cm）的内置 SATA 硬盘
尺寸规格	便携式存储设备最大尺寸为 14 英寸高、19 英寸宽、36 英寸长（相当于标准 19 英寸机架的 8U 大小）
重量	便携式存储设备的最大重量为 50 磅
设备容量	16 TB，更大的需求，请给 AWS 发信联系（awsimportexport@amazon.com）

表 10-3 欧盟 Region 的便携式存储设备要求

项目	需求
设备的外部电源要求	英/爱尔兰式 BS1363 插头、欧洲大陆 Schuko（CEE7）插头、或 Kettle Lead IEC320/C14 插头（其它形式插头请附寄转换器） 230 伏 / 50 赫兹 最大电源功率 2,000 瓦
接口类型	eSATA USB 2.0 或 3.0 （邮寄包裹需包含 USB flash 驱动） 2.5（6.4cm）或 3.5 英寸（8.9cm）的内置 SATA 硬盘
尺寸规格	便携式存储设备最大尺寸为 35 厘米高、48 厘米宽、91 厘米长（相当于标准 48 厘米机架的 8U 大小）
重量	便携式存储设备的最大重量为 22.5 公斤
设备容量	16 TB，更大的需求，请给 AWS 发信联系（awsimportexport@amazon.com）

表 10-4 亚太 Region 的便携式存储设备要求

项目	需求
设备的外部电源要求	230 伏 / 50 赫兹 最大电源功率 2,000 瓦
接口类型	eSATA USB 2.0 或 3.0 （邮寄包裹需包含 USB flash 驱动） 2.5（6.4cm）或 3.5 英寸（8.9cm）的内置 SATA 硬盘

项目	需求
尺寸规格	便携式存储设备最大尺寸为 35 厘米高、48 厘米宽、91 厘米长（相当于标准 48 厘米机架的 8U 大小）
重量	便携式存储设备的最大重量为 22.5 公斤
设备容量	16 TB，更大的需求，请给 AWS 发信联系（awsimportexport@amazon.com）

对导入 EBS 的情形，如果存储设备不大于 1 TB，则数据内容直接加载到一个 EBS 快照中。如果存储设备容量超过 1 TB，则会在一个特定的 S3 bucket 中存储该设备的镜像文件，然后用户可使用诸如逻辑卷管理器（Logical Volume Manager）之类的软件创建若干 EBS 卷组成的 RAID，并把镜像文件从 S3 拷贝到新的逻辑卷上。

对 S3 导入导出，支持的文件系统包括 NTFS、ext 及 FAT32，此外，也支持 HFS 以导入最大 2 TB 的分区。

AWS 通常支持更大或更重的便携式存储设备，这类请求需要和 AWS 联系解决，请在发送的邮件中说明 AWS 账号、要加载的 TB 数以及想要使用的设备类型。

AWS 数据中心的每个 AWS 导入导出工作站都可以每秒超过 100 MB 的速率加载数据，但具体数据加载速率受限于设备支持的读写速率，对 S3 的数据加载，还与 object 的平均大小相关。所以，选择高速读写接口的设备将节省数据加载时间。

虽然 AWS 导入导出服务是为具有 USB 或 eSATA 接口的存储设备而设计，但对 LaCie、MicroNet、Seagat、Western Digital 等设备一般来说也有不错的性能。

10.5 AWS Storage Gateway

10.5.1 简介

AWS 存储网关（AWS Storage Gateway）[59]作为一个 Web Service，将企业内部软件应用和云端存储相连，以在用户组织内部的 IT 环境和 AWS 的存储架构之间提供无缝且安全的集成方式。AWS 存储网关使用户可以安全上传数据到 AWS 云端，以达到成本经济的备份与快速灾难恢复的目的。AWS 存储网关支持工业标准级别的存储协议，以支持用户现有的应用。用户数据保存在内部存储硬件上，并低延迟地异步上传到 AWS，加密存储在 S3 中。

借助 AWS 存储网关，用户可以生成内部应用数据的快照，并备份到 S3。然后在任何时候，使用 AWS 存储网关将内部应用数据镜像到 EC2 实例中。

AWS 存储网关的使用非常简单，步骤如下：

1) 从 AWS 管理控制台下载 AWS 存储网关的虚拟机（VM）镜像文件。
2) 从直连存储（Direct Attached Storage，DAS）磁盘或存储区域网络（Storage Area Network，SAN）磁盘为安装完成的网关分配磁盘空间。AWS 存储网关在分配的磁盘上维护内部应用数据以提供低延迟的数据访问。用户可从新磁盘或已有数据的磁盘开始本步操作。
3) 使用 AWS 管理控制台来激活网关，将网关的 IP 地址和用户的 AWS 账号绑定，并

为网关选择一个 AWS Region 来存储上传的数据。

4) 使用 AWS 管理控制台创建 AWS 存储网关的卷，并把这些卷作为 iSCSI 设备连接到企业内部应用服务器上。
5) 用户的内部应用此时即可向第 4)步中连接上的网关存储卷写入数据，网关会在本地维护一份数据以保证低延迟的访问，同时异步向 S3 上传一份数据。

10.5.2 服务亮点

AWS 存储网关具有如下服务亮点：

- 安全：AWS 存储网关通过 SSL 将用户数据传到 AWS，存储在 S3 上的数据采用 AES 256 算法加密，AES 256 算法是一个采用 256 位加密密钥的对称密钥加密算法。
- S3 支撑：AWS 存储网关将用户内部应用数据以 EBS 快照方式存储到 S3 上。
- 兼容：AWS 存储网关对外提供了标准的 iSCSI 接口与用户已有应用交互。
- 与其它 AWS 集成：AWS 存储网关和 EC2、S3、EBS 无缝集成，其将用户数据以 EBS 快照方式存储在 S3 上。在需要时，可将此 EBS 快照恢复到 EBS 卷上，并连接到 EC2 实例上。
- 网络性能优化：AWS 存储网关只上传变化的用户数据，以减少数据上传备份的时间。用户也可使用 AWS 直连来提高网络吞吐率。

10.5.3 特点

AWS 存储网关具有如下特点：

- 网关存储卷：网关存储卷使用户数据在拥有可靠的异地备份的同时，用户内部应用仍可低延迟的访问全部数据集。存储卷最大可达 1 TB，可作为 iSCSI 设备 mount 到用户的内部应用服务器上。写到网关存储卷上的数据先存储在本地存储硬件上，然后异步地以 EBS 快照的形式存储到 S3 上。
- 数据快照：AWS 存储网关可创建网关存储卷的 EBS 快照并存储到 S3 上，用户可随时创建快照，也可设置定期创建。快照都是增量备份，而且所有快照都是压缩后存储的。
- 计量：可使用 Amazon CloudWatch 来监控网关存储卷的性能参数，包括用户内部应用的吞吐率、延迟、到 S3 的 Interent 带宽等，这些计量参数可从 AWS 管理控制台上访问到。
- 网关缓存卷：Amazon 将很快提供网关缓存卷的功能。写到网关缓存卷的数据将被存储到 S3 上，只在本地存储硬件上保留最近写入数据以及经常访问数据的缓存。用户可以将对延迟要求不高的数据存储到 S3 上，而把要求低延迟访问的数据存储在本地，这样，在减小用户扩展本地存储框架的压力的同时，仍然可以为用户应用提供活跃数据的低延迟访问。

10.5.4定价

注册后，AWS 存储网关会提供一个 60 天免费使用的网关。此外，作为 AWS 免费使用包的一部分，用户还可以在一年获得如下免费：每月 1 GB 存储空间、每月 15 GB 从所有

AWS 服务的累计数据输出量。超出以上部分时，AWS 存储网关的费用与网关、存储和数据传输量有关[59]。

- 网关：每月每个激活的网关收费 125 美元。
- 存储：指 EBS 快照在 S3 上的存储费用，各个 Region 费用稍有不同，大致为每月每 GB 收取 0.13 美元。
- 数据传输量：由 AWS 存储网关传入的数据量免费，传出的数据每月前 1 GB 免费，之后每月 10 TB 的每 GB 收费 0.12 美元（Region 不同价格有所区别），更多数据量的价格请参考 AWS 存储网关主页。

10.5.5 常用场景

AWS 存储网关具有广泛的应用场景，包括：

- 灾难恢复
  用户数据以 EBS 快照形式存储在 S3 上，S3 会将其上的快照在多个场所的多台设备上冗余存储，并定期校验，快速检测并修复任何丢失的备份数据。当意外发生导致用户本地基础架构宕机时，用户可以启动 EC2 实例，并将存储在 S3 上的快照恢复到新的 EBS 卷上，再将该卷连接到刚运行的 EC2 实例上，即可实现灾难恢复。
- 异地备份：AWS 存储网关将用户本地应用数据异地备份到 S3 上，所有数据通过 SSL 传输并采用 AES 256 加密算法加密。AWS 存储网关是多数据中心备份以及磁带备份的不错替代选择。
- 数据镜像到云端计算资源：如果用户想在峰值期间或在新的项目中利用 EC2 的按需扩展能力，可以使用 AWS 存储网关将本地数据镜像到 EC2 实例上。如果用户在 EC2 上跑 User Acceptance Test（UAT）环境，可以使用 AWS 存储网关把用户本地生产系统上的最新数据镜像到 EC2 上 UAT 环境中。所谓镜像操作，即使用 AWS 存储网关将用户应用数据以 EBS 快照方式上传到 S3 上，并使用 AWS 管理控制台或 EC2 API 从这些快照创建 EBS 卷，然后将这些 EBS 卷连接到 EC2 实例上，一旦连接完成，EC2 实例就可随时访问这些数据。

10.5.6 产品细节

1) 硬件要求

AWS 存储网关的虚拟机（VM）必须安装在至少满足如下要求的主机上：

- VMWare ESXi Hypervisor（v4.1 或 v5），可从 VMWare 网站获取免费版本[118]。
- 该 VM 需要 4 虚拟核。
- 该 VM 需要 7.5 GB 内存。
- 该 VM 需要 7.5 GB 磁盘空间，用于.ova 文件安装和系统数据存储。

目前支持使用 Microsoft Windows 或 Red Hat iSCSI Software Initiator 来 mount AWS 存储网关的存储卷。

2) AWS 存储网络详情

AWS 存储网关的部署如图 10-1 所示：

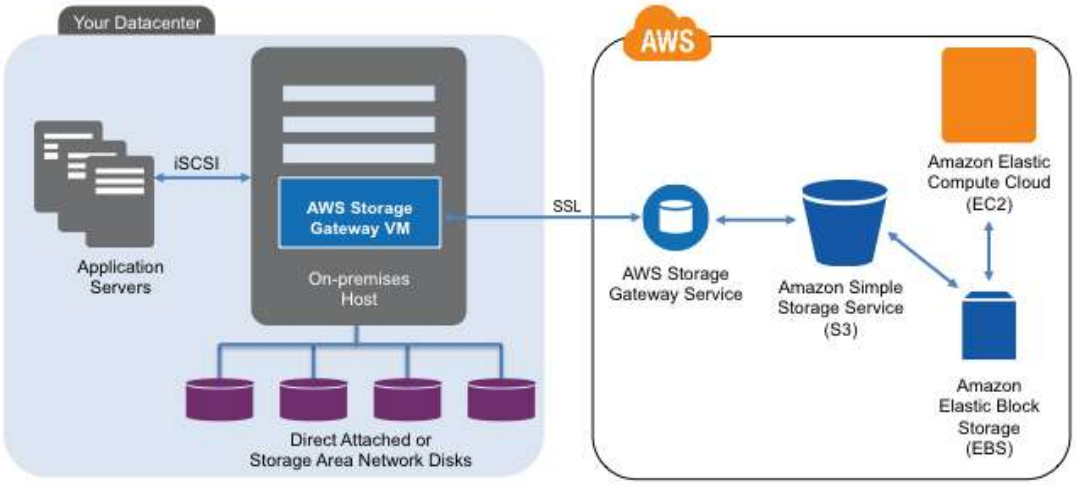

图 10-1　AWS 存储网关部署图

用户完成安装流程后，AWS 存储网关的软件 appliance 就已被安装到用户数据中心的某台主机上，此时可以指定用来存储备份数据的 AWS Region，并为用户的 AWS 账号绑定 IP 地址以激活网关。然后，用户可以创建网关存储卷，并把这些卷映射到本地的 DAS 或 SAN 磁盘上，这些磁盘可以是新的磁盘也可以是已经包含数据的磁盘。之后，把这些存储卷作为 iSCSI 设备 mount 到用户本地的应用服务器上。当用户本地应用从网关存储卷读取或写入数据时，数据会从本地 DAS 或 SAN 磁盘上获取或写入。

在数据上传至 S3 之前，网关 VM 会将入口数据缓存到一个临时区域，称为工作存储区（Working Storage），本地的 DAS 或 SAN 磁盘可作为工作存储区。网关 VM 将工作存储区中的数据通过加密的 SSL 连接上传到 AWS 云端运行的 AWS 存储网关服务上，再由 AWS 存储网关服务将数据加密存储到 S3 上。

打快照时，AWS 存储网关会把网关 VM 的工作存储区中自最近一次快照后所有变动的数据上传到 S3，并存成 EBS 快照形式。快照是增量备份，可以定期或随时触发打快照操作。

当需要从备份中恢复数据时，可以将 EBS 快照恢复到本地的网关存储卷中，或者将快照恢复回一个新的 EBS 卷，并将该其连接到 EC2 实例上。

第十一章 支持类（Support）

11.1 AWS Support

11.1.1 简介

AWS Support[119]提供了专业技术支持工程师的一对一 24x7x365 的快速支持渠道，帮助各类客户和开发者成功地使用 AWS 的产品和特性。

11.1.2 特点

AWS Support 提供了四种级别的个性化技术支持服务，其中，基本级别的支持是免费的，包括资源中心[120]、产品 FAQ[121]、论坛[122]及健康检查。AWS Support 服务内容详见表 11-1。

表 11-1 AWS Support 服务内容

	基本	开发者	商业	企业
24x7x365 客户服务	✔	✔	✔	✔
论坛	✔	✔	✔	✔
文档、白皮书、最佳实践指南	✔	✔	✔	✔
技术支持	健康检查	Email（本地工作时间）	电话、聊天、Email(24/7)	电话、聊天、Email、电话答录机(24/7)
指定联系人	/	1	5	无限
反馈时间	/	12 小时	1 小时	15 分钟
架构支持	/	模块级别	用例指导	应用架构级别
最佳实践指导	/	✔	✔	✔
客户端诊断工具	/	✔	✔	✔
资深支持工程师支持	/	/	✔	✔
第三方软件支持	/	/	✔	✔
AWS 可信导师	/	/	✔	✔
基础架构事件管理	/	/	收费服务	✔
用户技术经理支持	/	/	/	✔
优质案例指导	/	/	/	✔

	基本	开发者	商业	企业
对管理的商业审核	/	/	/	✔

11.1.3 定价

所有级别的 AWS Support 服务都包括数目不限的案例支持。对商业和企业级支持，随着 AWS 使用费用的上升，还有更多折扣[119]，如表 11-2 所示。

表 11-2 AWS Support 价格

AWS Support 级别	价格
基本	包含在 AWS 产品价格中
开发者	每月 49 美元
商业	每月高于 100 美元，或：每月 AWS 费用为 0-1 万美元时，收取 10%；1 万到 8 万时，收取 7%；8 万到 25 万时，收取 5%；超过 25 万时，收取 3%
企业	每月高于 15000 美元，或：每月 AWS 费用为 0-15 万美元时，收取 10%；15 万到 50 万时，收取 7%；50 万到 1 百万时，收取 5%；超过 1 百万时，收取 3%

第十二章 Web 流量类（Web Traffic）

12.1 Alexa Web Information Service

12.1.1 简介

Alexa Web 信息服务（Alexa Web Information Service，AWIS）[123]为开发者提供 Alexa 的有关流量和 Web 结构的大量信息。

12.1.2 特点

AWIS 具有如下特点：

- 可收集网站的相关信息，包括流量数据、联系信息、相关链接等。
- 访问网站的历史流量数据，来分析网站流量的增长趋势以及特殊事件对流量的影响。
- 可使用 Alexa 增强版的基于 DMOZ[124]的浏览服务，在用户网站或服务中构建一个 Web 目录。
- 能访问链接到指定网站的网站列表。

12.1.3 定价

按使用计费，每 1000 个请求收取 0.15 美元。

12.1.4 产品细节

AWIS 提供了如下操作，或称为“动作（action）”：

1) URL 信息

   Alexa 通过其 Web 爬虫和 Web 使用分析收集了大量有关网页和网站的信息，URL 信息（URL Information）动作使开发者可以直接访问这些信息，这些信息包括网站流行度、相关网站、详细使用/流量统计、支持的字符集/区域设置、网站联系信息等。这是在 Alexa 网站及 Alexa 工具条[125]中最常见的数据，以及部分随 AWIS 而首次发布的信息。

2) 历史流量

   历史流量（Historical Traffic）动作使开发者能够通过编程方式，来访问过去一年的网站流量排名、每百万人中访问人数（reach）、页面访问量信息。可以使用该动作比较一段时间内网站流行度的变化、确认趋势，或者展现流量图表。

3) 网站链接

网站链接（Sites Linking In）动作返回链接到指定网站的网站列表。

4) 流量分类

流量分类（Browse Category）动作允许开发者访问开放目录（Open Directory）下的所有可用信息，而无需将信息下载到本地系统或在本地系统中安装目录数据库。该服务返回某个特定分类下的网页及子类。返回的 URL 已被 Alexa 流量数据过滤过，并已按流行度排序。

12.2 Alexa Top Sites

12.2.1 简介

Alexa 顶级网站（Alexa Top Sites）[126]服务提供了对 Alexa 流量排名[127]网站列表的访问，开发者可使用该 Web Service 了解从流量最大到最小的网站排名。该服务每页展示 100 个网站，并可通过多次请求，获取 1000、5000 或 100000 个网站列表。

除了 Alexa 流量排名，返回的每个网站信息还包括每百万个人中访问数、每个用户平均页面访问量，以及 Alexa 用户访问数。

12.2.2 特点

Alexa Top Sites 服务具有如下特点：

- 按 Alexa 流量排名列出顶级网站。
- 数据基于全球或某个（些）国家。
- 可获取每个网站的流量排名、每百万人中访问数、页面访问量信息。
- 能使用户应用或服务感知网站的流行度。

12.2.3 定价

每 100 个 URL 收取 0.25 美元。

12.2.4 产品细节

Alexa Top Sites Web 服务提供了顶级 Interent 网站的排名列表。

Alexa 的流量排名是在最大且最全球化的一个用户样本上采用匿名使用模式而得到的，该 Web 服务使开发者能为其网站及应用添加高质量的 Web 流量信息。这些数据可内部使用来帮助作出商业决定，例如寻找最有效的广告投放地；也可外部使用，例如按优先级排序网站的聚集信息。开发者既可以查找全球最大的 100000 个网站，也可查找美国最大的 1000 个网站，Alexa Top Sites 允许开发者以编程方式获取来自 70 多个国家的网站排名。

网站排名是基于最近 3 个月的每百万人中访问数以及页面访问量的组合计算而得。每百万人中访问数取决于某天访问某个网站的 Alexa 用户数，页面访问量是 Alexa 用户对某个网站的总页面请求数。但是，同一天同一用户对同一页面（URL）的多次请求计为一次页面访

问。用户和页面访问量组合最高的网站即为第一名。

流量排名只按顶级域名排列（例如，domain.com），域名下的子页面（例如，www.domain.com/subpage.html）或子域名（例如，subdomain.domain.com）不单另排名，除非它们是个人主页或博客。

第十三章 人力类（Workforce）

13.1 Amazon Mechanical Turk

13.1.1 简介

即便在信息化技术高度发展的今天，人类智力仍然在相当多的方面表现得比计算机更为迅捷有效，如图像或视频识别、数据去冗余、录音誊抄等。对这些计算机处理起来既耗时昂贵又难于扩展的任务，雇用一大批人力来做可能更为经济有效。Amazon Mechanical Turk[1]即是这样一个提供了人力智能劳务交易的市场，其目标是简单、可扩展、高性价比地使用人力智能资源。AMT 由 Amazon MTurk、Requester 和 Worker 三部分组成，其应用流程为，Requester 在 AMT 平台上发布劳务信息（即人力智能任务，Human Intelligence Tasks, HITs），Worker 选择任务，并可在完成后获取报酬。AMT 平台提供了 API，使得 Requester 可以编程访问平台、定义任务并将任务结果直接集成到自己的应用中[1]。

13.1.2 特点

AMT 提供一个一种按需使用人力的弹性机制，同时结合质量控制逻辑来帮助有人力智能需求的用户降低其任务成本，并可能催生新的业务模式。

13.1.3 定价

按用付费，Requester 支付给 Worker 的费用的 10%将被 AMT 以佣金方式收取，每个 HIT 的佣金下限是 0.005 美元。

13.1.4 常用场景

- 图像/视频处理：如标注图像中的特定对象以便于搜索、从一组图片中选择最能代表某个产品的图片、审查用户上传的图片是否含有不合适的内容、分类卫星拍摄的图像中的物体等；
- 数据校验与清理：去除黄页目录列表中的冗余数据、发现在线产品目录中重复条目、验证某些详细信息（如餐馆的电话号码、营业时间等）等；
- 信息收集：允许用户提出任何问题，并雇佣 Workers 回答这些问题、完成一个内容复杂的调查、为网站撰写评论/描述/blog、从法律和政府文件中查找特定的信息等；
- 数据处理：Podcast 内容的编辑和誊抄、人工翻译、评价搜索引擎结果的精确度等；

13.1.5 运营分析

AMT 从 2005 年 11 月上线至今仍然是 Beta 版本，相较其他的 AWS，表现得并不突出，究其原因，主要有如下几点：

- 用户的接受参与度有待提高，相当多的 HITs 仍然是 Amazon 自己发布的，如评论 CD、书籍等；
- 2010 年底的一次调查显示，AMT Worker 所提供的结果中 40.92%为垃圾结果[128]，这就迫使 Requester 试图一开始就雇佣更多的 Worker 来完成任务以保证质量，其直接结果就是导致每个 HIT 的报酬下降；
- 另一方面，有些 Requester 在收到可以接受的任务结果后，却试图通过报垃圾结果的方式来逃避付酬；

由此可见，AMT 业务有几个需要妥善处理的关键点：

- 便捷的任务质量验证、控制与评价体系；
- Worker 的奖惩与优胜劣汰机制；
- 交易双方的信用机制与第三方监管；

第十四章 基于 AWS 的解决方案

本章将介绍基于 AWS 的 12 类解决方案，覆盖 Web 应用、企业应用、大数据及高性能计算、灾难恢复及归档、公共部门等领域。

14.1 Web 应用

本节讨论 Web 应用，包括移动及社交应用等，使用 AWS 来构建底层的 IT 框架，从而实现快速部署和经济运行的目的。

一个基于 AWS 的 Web 应用架构的典型例子如图 14-1 所示[129]。

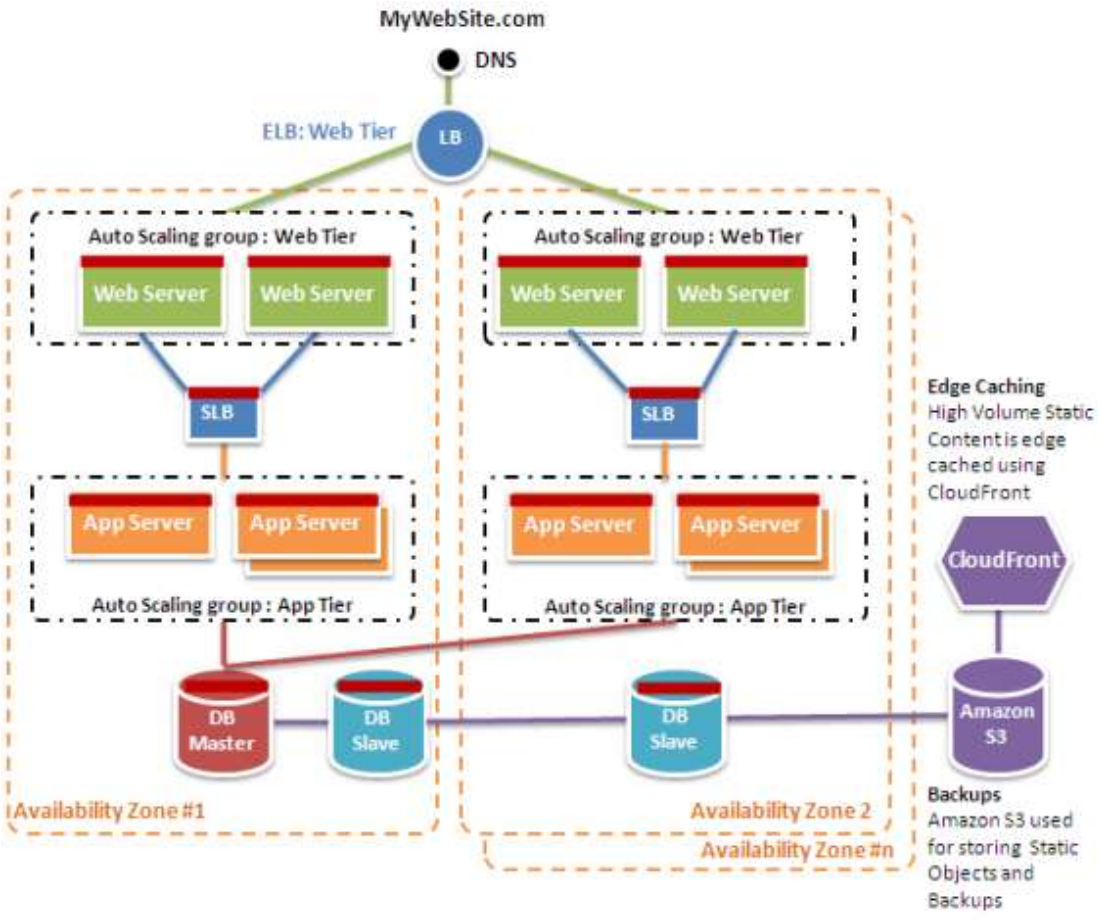

图 14-1 基于 AWS 的 Web 应用架构例子

可使用 AWS 服务来支持 Web 应用各种需求，如表 14-1 所示[130]。

表 14-1 AWS 支持 Web 应用

动态内容支持	Amazon EC2 提供 Linux 和 Windows 两种类型虚拟机，用户可在其上部署 Web 或应用服务器，并可使用 Auto Scaling 和 ELB 来无缝扩展，以支撑任意数量用户的访问。
存储并访问应用的数据集	AWS 的 DynamoDB 为用户提供了高性能的可应对任意扩展的 NoSQL 数据库服务。DynamoDB 适合表有一个主键、对延迟敏感且 I/O 密集的应用。用户还可使用 Amazon RDS 服务来使用高可靠性的 MySQL 或 Oracle 数据库服务，满足事务性或需要复杂关系型查询的应用需求。
按需的流或实时内容分发	借助 AWS 的全球内容分发网络及 CloudFront 服务，用户可以将自己的内容快速分发到客户，加快页面的访问速度。
运行应用软件	用户可使用 Amazon EC2 来运行自己的软件或第三方的软件，并可运行自己的负载均衡软件来扩展应用需求。
存储静态 Web 或应用内容	用户可使用 Amazon S3 来存储任意数量的 Web 或应用静态内容，并可为这些内容设置开放权限，从而可从任意地点访问这些内容。
将客户路由到用户的应用	借助 Amazon Route 53 的 DNS 服务，可以保证客户的请求能被快速地路由到用户的应用。
个人化应用	Amazon Elastic Map Reduce 提供受管的 Hadoop 服务，使用户能够简便地创建并优化算法，以加强用户 Web 应用的个性化特点。

14.2 游戏

游戏公司可以在云端运行游戏服务器，来降低成本、快速部署并更易于扩展。AWS 云服务能在分钟级别部署上千台虚拟机，简单快速地相应社交游戏的高扩展需求。Facebook 上的 10 大最流行游戏中已经有 8 个部署在 AWS 之上[131]。

下面以 Facebook 的游戏应用为例，讨论基于 AWS 的应用架构，如图 14-2 所示[132]。

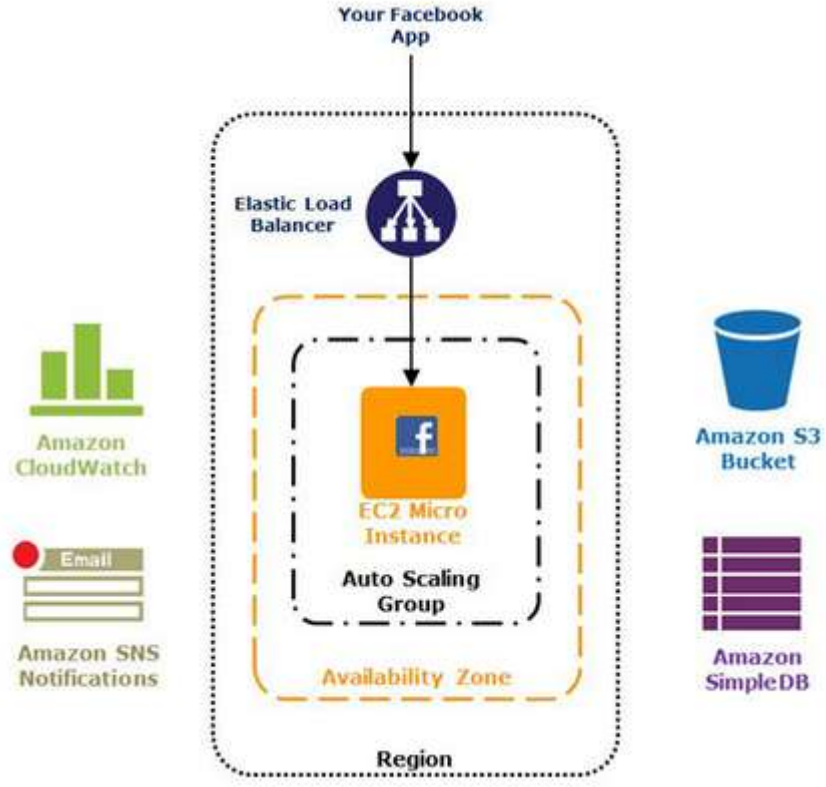

图 14-2　基于 AWS 的 Facebook 应用架构

Facebook 基于 AWS CloudFormation 提供了样例模板，帮助用户快速在 AWS 上部署 Facebook 应用，包括游戏或社交类应用。这个模板用到了 EC2 实例来运行应用，前端通过 ELB 提供负载均衡能力，文件等可存放在 S3 上，应用数据存储在 SimpleDB 中，通过 CloudWatch 来监控 EC2 实例及应用的运行状况，SNS 提供 Email 级别的告知服务。

除了 Facebook 的样例模板，用户也可根据自己游戏的实际情况，选用不同的 AWS 服务，例如，如果游戏用户分布在全球不同地理区域，可以选用 AWS CloudFront 加快内容分发，提高游戏体验。用户还可使用 RDS 将应用数据存储在 MySQL、SQL Server 或 Oracle 数据库中以满足事务性或复杂关系型查询的应用需求，架构如图 14-3 所示[133]。

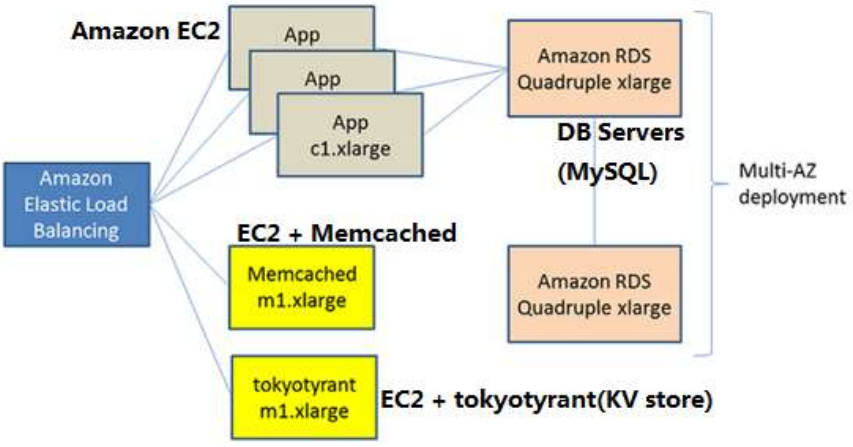

图 14-3　另一种基于 AWS 的游戏应用架构

14.3 媒体分享

商业应用可以使用 AWS 云服务来运行照片、视频或文件存储应用，这种方式可以大幅降低部署及运行成本，提高上线速度并实现存储的高扩展性。Amazon S3 可以存储上亿个对象，在任意时间从 Web 任何地点来读取或存储任意大小的数据。S3 提供年 99.999999999%的数据可靠性与 99.99%的可用性[134]。

基于 AWS 的一个媒体分享应用的框架设计如图 14-4 所示，这是 photoWALL 公司基于 AWS 的架构设计，提供社交媒体、实时 Web 应用及移动终端应用的服务[135]。

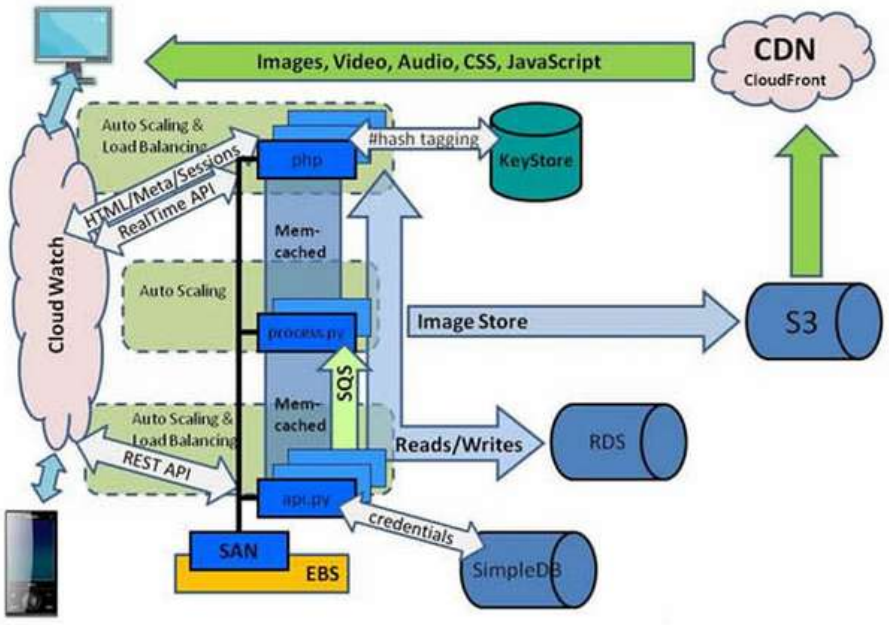

图 14-4　photoWALL 的基于 AWS 的应用架构

图中，一个超过 25 台 EC2 实例组成的集群作为 photoWALL 的 API 服务器、照片处理服务器及 PHP 应用，此 EC2 实例集群采用 Amazon ELB、Auto Scaling、CloudWatch 及 EBS 服务提供负载均衡、自动扩展、应用监控及 EC2 的块存储能力。Amazon SQS 管理应用工作流，一旦某张图片通过验证，即发送到照片处理服务器进行处理（如调整大小、加位置标签、建索引等），SQS 还可用来将照片排队并发送到支持的社交媒体网站。照片的元数据信息主要存储在 Amazon RDS 中（采用 MySQL 数据库，并使用 memcached 加速查询），出于冗余备份的考虑，某些元数据还会在 Amazon SimpleDB 中存储一份。照片处理完成后，会存储到 Amazon S3 上，并通过 CloudFront 分发到终端客户。

此外，在 EC2 上的图片处理过程中，可以基于 Amazon Elastic MapReduce 服务，使用用户的 Pig 脚本及 Hadoop Java 程序来完成批量处理工作。

14.4 企业级应用

各种类型的企业都可以在云上运行自己的商业应用，以降低成本、快速部署并简化管理框架。用户既可在 AWS 之上部署自己的商业应用，也可完全采用 Oracle、SAP、Microsoft 和 IBM 的软件栈，AWS 提供了大量预配置的虚拟机镜像文件，可以在分钟级别启动这些软

件栈。用户还可以在 AWS 上使用其已有的软件许可，而无需缴纳额外的费用。

除了部署应用外，用户也可以订购 Adobe、Infor、ESRI、Splunk、Consona、Coupa 等 SaaS 软件厂商提供的运行在 AWS 上的商业应用[136]。

AWS 服务可有效支持企业级应用在云上的部署与运行，表 14-2 列举了如何使用 AWS 服务来满足企业级应用的需求。

表 14-2　AWS 服务支持企业级应用

连接已有数据中心	用户可以使用 AWS 直连与 Amazon VPC 服务，在用户现有的数据中心和 AWS 云之间建立无缝连接，并在两者间迁移工作负载。
支持用户扩展	借助 Amazon EC2、Auto Scaling 和 ELB 服务，可以自动增加数以千计的虚拟机，以实现无缝的扩展，来满足任意数量的用户访问需求。
数据与应用的迁入迁出	在 AWS 中，用户可选择任意操作系统、编程语言或数据库来满足自己商业场景的需求。AWS 还提供了导入导出服务及虚拟机导入功能，帮助用户迁移数据和虚拟机镜像文件。
建立企业级的安全标准	AWS 具有工业级认证与审计资质，包括 SOC 1 Type 2，ISO 27001，PCI DSS，FISMA Moderate，HIPAA，ITAR 等。
为全球化企业部署全球化体系架构	AWS 在全球部署云计算服务的基础架构，AWS Region 遍布美、欧、亚洲，而且 AWS CloudFront 在全球部署有众多节点服务器，使用户数据能快速分发到终端客户。

下面以美国复苏问责与透明度委员会（The Recovery Accountability and Transparency Board，RATB）的 Web 平台为例介绍基于 AWS 的企业应用架构方案[137]，如图 14-5 所示。

2009 年 2 月，美国国会通过美国复苏与再投资法案（ARRA）后，成立了 RATB，以确保不会发生浪费、欺诈、滥用的现象。RATB 需要架构一个网站（www.recovery.gov）来完成如下目标：

- 使公众能否方便地获取复苏资金的使用情况和结果，并公开问责
- 将官方数据交由公众讨论
- 公平公开地竞争经济复苏中的机会

RATB 目前使用了如下 AWS 服务：Amazon EC2 上运行网站应用以及其它企业级应用，如 Microsoft SharePoint、SAP BusinessObjects、Microsoft SQL Server 等，Amazon ELB 在 EC2 集群上提供负载均衡服务，Amazon EBS 为 EC2 提供额外的块存储能力，Amazon S3 提供海量数据的存储服务，Amazon CloudWatch 监控整个网站应用及服务的健康情况。

包括 Oracle[138]、SAP[139]、Microsoft[140]、IBM[141]在内的大量企业级应用软件已经与 Amazon 合作，推出了大量定制的 AMI，用户可参考各个公司的 AWS 迁移指南，选择合适的 AMI，以在 Amazon EC2 上启动相应的企业级应用软件。

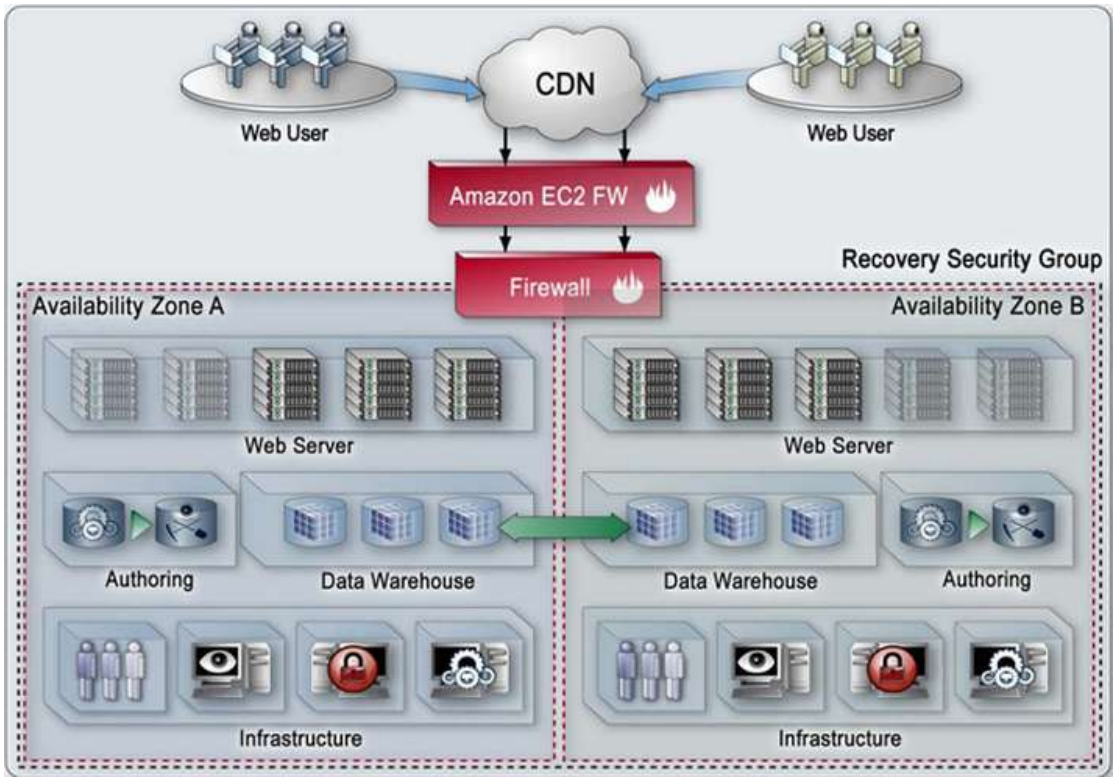

图 14-5　基于 AWS 的 RATB Web 平台架构

14.5 大数据

使用传统的 IT 技术处理大数据会感到力不从心，因此新的技术被用来解决大数据问题，如 Hadoop 框架等。AWS 提供了一组服务来高效经济的处理大数据并提取信息，如 Amazon Elastic MapReduce 等[142]。

用户可混合使用高内存和高 CPU 的 Amazon EC2 实例，并在其上搭建 Amazon Elastic MapReduce 集群，分析数据存储在 Amazon S3 中。

14.6 高性能计算（HPC）

通过 EC2 实例类型的集群支持，可以基于 Amazon EC2 搭建 HPC 的解决方案。EC2 的 Cluster Compute 和 Cluster GPU 实例可用来优化 HPC 应用：

- 可在 Placement Group 中启动集群实例，在 Placement Group 中启动的所有实例都具有低延迟、实例之间全双工 10 Gbps 带宽连接。Placement Group 是动态并可按需扩展。在大规模并行处理中，还可连接多个 Placement Group 来建立超大的集群。
- Cluster Compute 和 Cluster GPU 实例在定义中指定了处理器的架构，开发者可以根据指定的处理器架构来调整其应用，以优化性能。
- 开发者可以利用 Cluster GPU 实例的 NVidia Tesla GPU 并行性能优势，采用 CUDA 及 OpenGL 编程模型来优化 GPGPU 计算。

利用 Amazon EC2 集群实例的先进网络及高计算能力，开发者可以获得超级计算类的能力，例如，由 1064 个 cc2.8xlarge 实例（17024 核）组成的集群可以获得 High Performance Linpack[143] benchmark 的 240.09 TeraFLOPS 的表现，这在 2011 年超级计算机 Top 500 列表中能排到第 42 位。

开发者可以并行多个集群来突破应用扩展的限制，还可采用 EC2 Spot 实例来显著降低成本开支。

14.7 灾难恢复

基于 AWS，企业可以实现快速的灾难恢复能力。AWS 支持很多流行的灾难恢复架构，从较小的易于扩展的 pilot light 环境到可以快速 failover 的 hot standy 环境。AWS 拥有遍布全球 8 个 Region 的数据中心，使用户 IT 基础架构及数据的快速恢复成为可能。

一个基于 AWS 的数据备份例子如图 14-6 所示[144]。

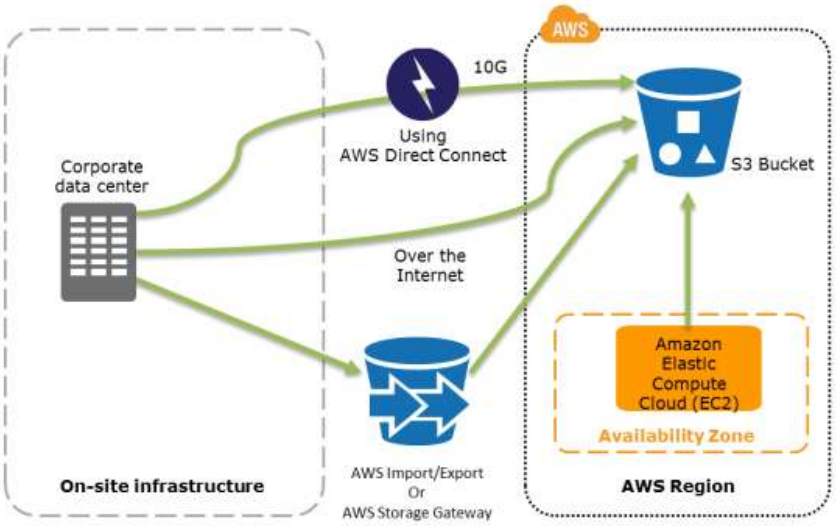

图 14-6　从企业数据中心或 AWS 向 Amazon S3 备份数据的例子

Amazon S3 是一个理想的数据备份选择，其提供年 99.999999999%的数据可靠性。对于用户数据中心的数据，可以通过 Internet 或 AWS 直连方式备份至 S3（或直接将应用迁移到 AWS VPC），对海量数据还可使用 AWS 导入导入服务。用户亦可采用 AWS 存储网管服务将本地数据中心的快照数据透明地备份到 S3 上，之后，用户就可以创建本地卷或 AWS EBS 卷来恢复这些快照。对于运行在 AWS 上的系统，数据可直接备份至 S3，EBS 卷的快照及 Amazon RDS 的备份亦可存储在 S3 上。无论何种情形，均可考虑使用 AWS SNS 在事件发生时通知用户。

另一个更为复杂的跨 Region 的灾备方案如图 14-7 所示[145]。

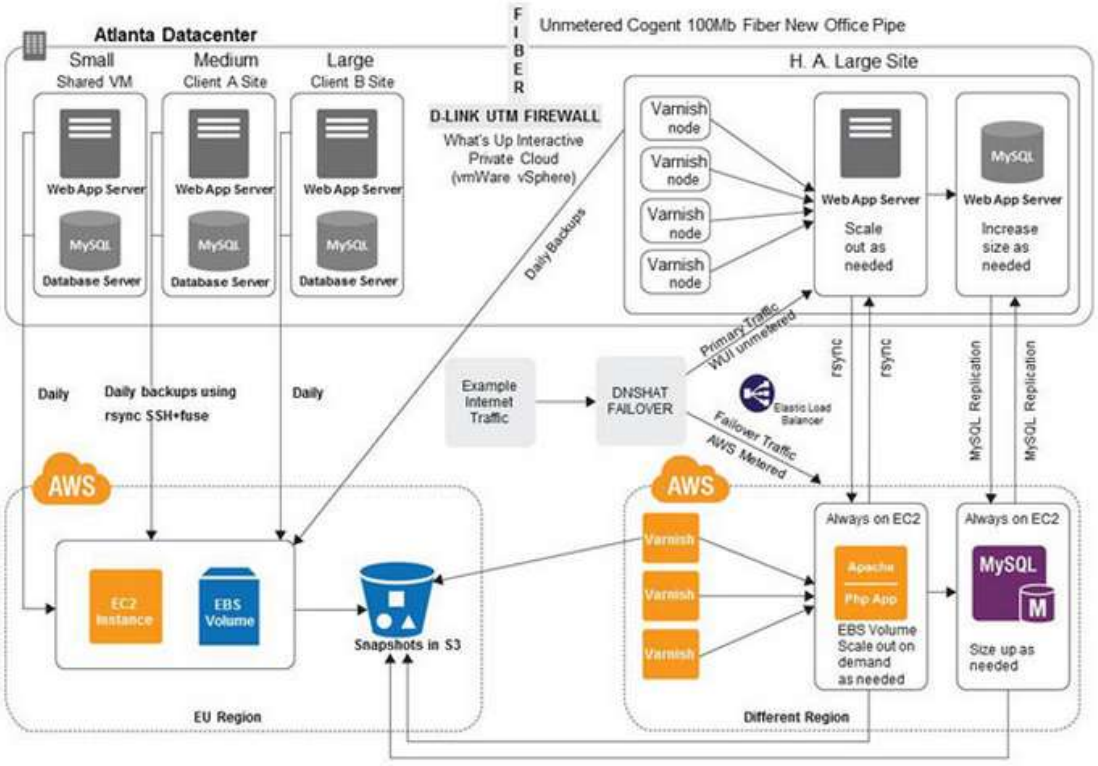

图 14-7　跨 Region 的灾备方案

恢复策略有多种情况，其中一种情况如图 14-8 所示。用户可以将保存在 S3 中的数据对象拷贝回 EC2 实例，例如 EBS 卷快照，并基于快照创建新的 EC2 实例。

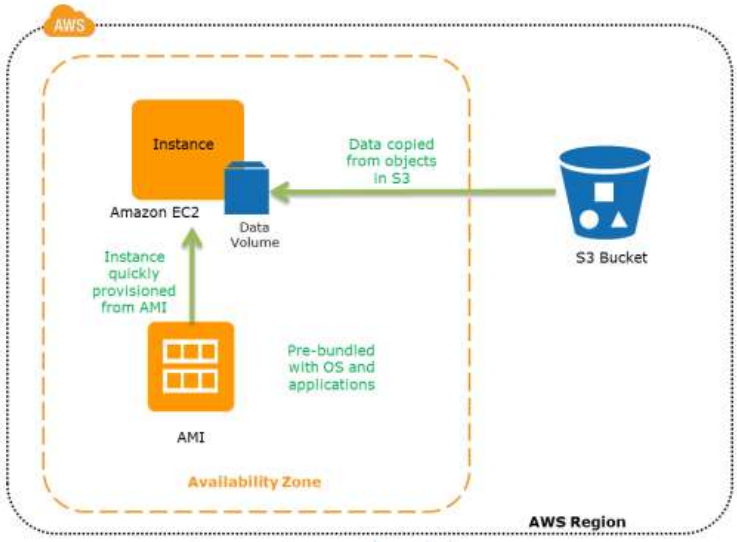

图 14-8　将 S3 中的备份数据恢复至 EC2

14.8 归档

AWS 提供了非常适合数据归档的服务，其中，Amazon Glacier 作为一种极其低廉的存储服务（每 GB 每月能低至 0.01 美元），提供了安全且可靠的数据归档和备份能力。Amazon

Glacier 适合那些归档后不经常访问的数据，并且能接受数小时的数据获取时间。Amazon S3 提供了高可靠且安全的长期存储服务，可用来归档用户数据，适合更频繁的数据访问以及要求即时获取数据的情况。

14.9 政府、教育等公共部门

美国联邦政府及其代理机构，包括美国国防部、NASA 等都在其业务逻辑中采用 AWS 服务。Amazon 也专门提供了一个 AWS Region（AWS GovCloud[19]），美国政府、代理结构及合同承包商可将敏感的工作负载迁移到 GovCloud 上，以满足特定的法规和审计要求。

除了政府部门之外，教育机构（包括各级大学、中学及培训机构等）也可采用 AWS 搭建各种云服务应用，架构设计与之类似。

我们先来看看 NASA 喷气推进实验室（Jet Propulsion Laboratory，JPL）的视频流服务的架构设计，如图 14-9 所示。

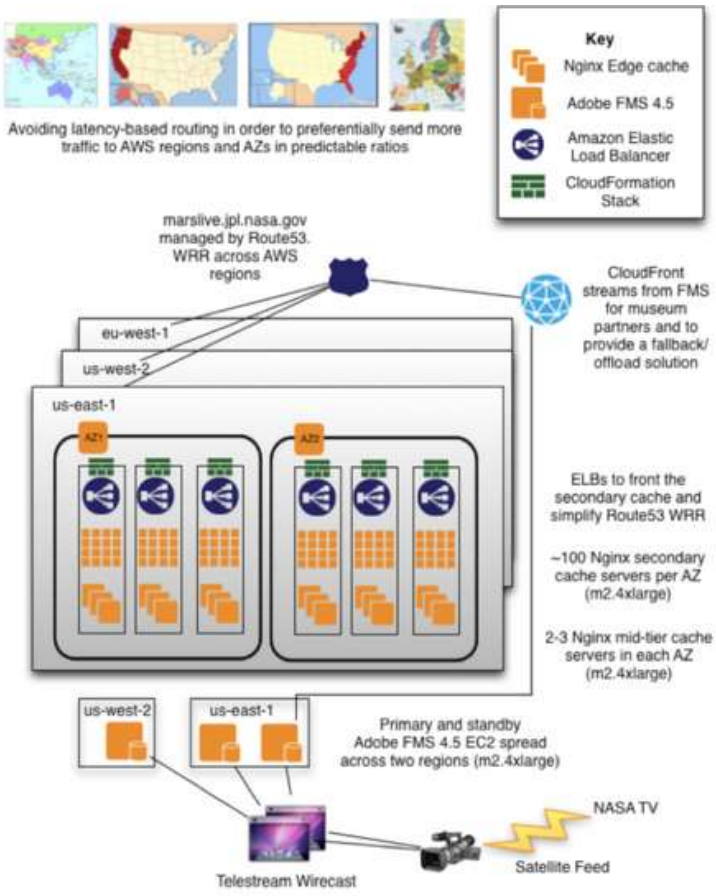

图 14-9　NASA JPL 视频流服务的架构

NASA JPL 的视频流服务架构基于 Adobe Flash Media Server 播放视频，Nginx 运行在 Amazon EC2 中，前端架设 Amazon ELB 提供负载均衡服务，同时配置 Amazon Route 53 来管理 DNS 路由，采用 Amazon CloudFront 分发视频内容，使用 AWS CloudFormation 在多个 AWS AZ 及 Region 间自动化地部署视频流服务架构的软件栈。

NASA JPL 的网站也采用了 AWS，其架构设计如图 14-10 所示。

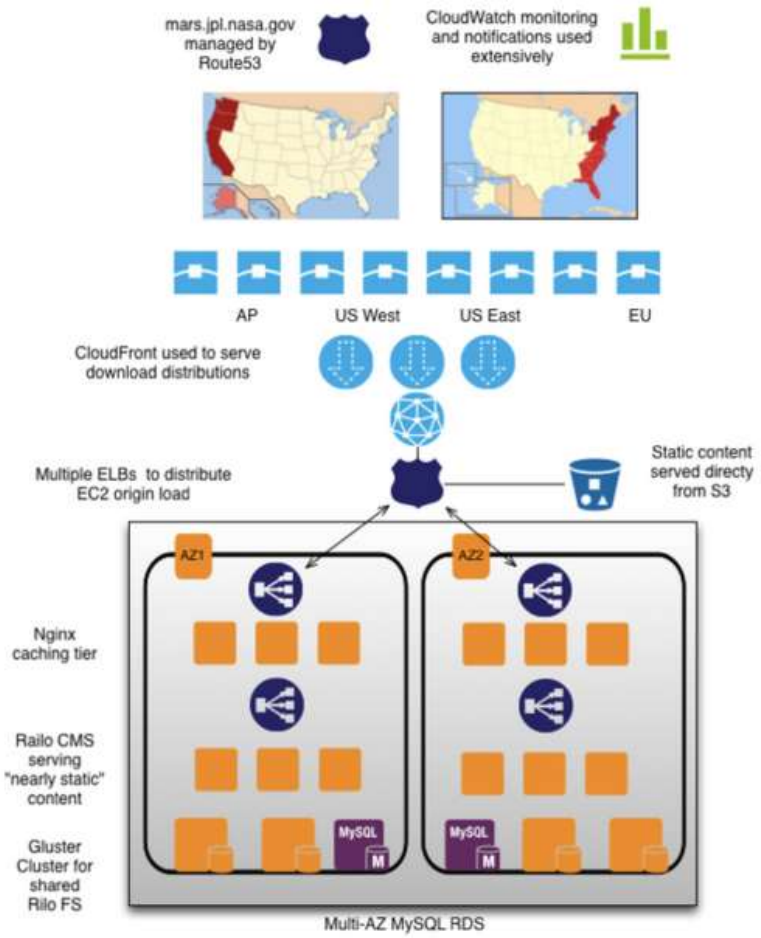

图 14-10　NASA JPL 网站架构

NASA JPL 的网站（mars.jpl.nasa.gov）是基于开源的内容管理系统 Railo 设计的，Railo 运行在 Amazon EC2 上，并配置 Amazon EBS 卷池来实现 EC2 实例的共享存储，Railo 还要与部署在多个 AZ 中的 Amazon RDS 的 MySQL 数据库交互。运行 Railo 的 EC2 集群前端配置了 Amazon ELB 及 Amazon Route 53 实现负载均衡与入口流量转发。Amazon CloudFront 用来在全球分发内容。此外，NASA 还采用了 Amazon SWF 将来自火星的最新图片拷贝至 Amazon S3，图片的元数据存储在 Amazon SimpleDB 中，每当“好奇号”有数据传输到地球时，Amazon SWF 就会触发工作流，部署 EC2 实例来处理收到的火星图片。

下面，我们再来看看美国国家可再生能源实验室（National Renewable Energy Laboratory）使用 AWS 的例子，该实验室的网站 OpenEI.org 采用了若干 AWS 服务，如图 14-11 所示。

OpenEI.org 采用 Amazon EC2 来运行 OpenEI 的所有关键软件组件，包括 Apache、Semantic MediaWiki、MySQL 及 OpenLink Virtuoso 等，采用 Amazon EBS 来存储 OpenEI 数据库，采用 Amazon S3 来备份所有 OpenEI 的数据，并在 EC2 实例集群前端部署 Amazon ELB 提供入口网络请求的负载均衡。

为提供端到端的安全和隐私保护，AWS 构建服务时充分考虑了客户授权、安全相关的最佳实践，并在服务中提供了合适的安全特性。AWS 遵守的安全框架包括 FISMA、PCI DSS、ISO 27001、SOC 1/SSAE 16/ISAE 3402 (以前称之为 SAS 70 Type II)，以及 HIPAA。

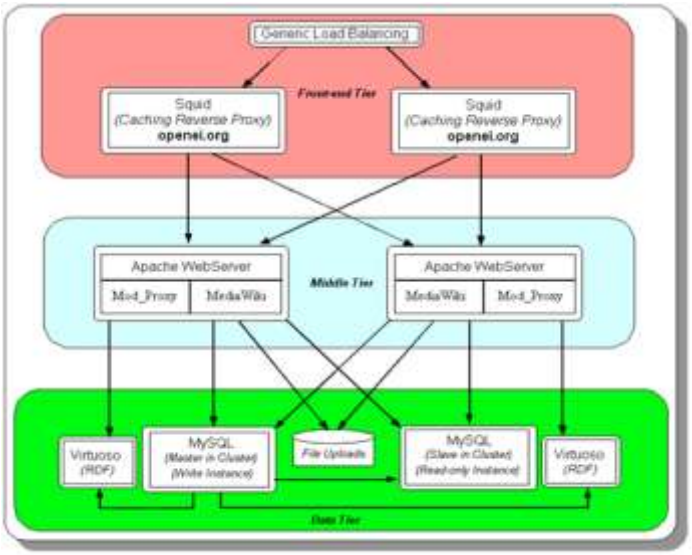

图 14-11　OpenEI.org 的网站架构

第十六章 AWS 未来发展小议

本节仅从作者的个人角度对 AWS 未来发展作一浅显的讨论。AWS 未来可能会从以下几个方向加强并发展：

1) 现有 AWS 服务的功能性增强

   随着互联网及云计算的进一步深入应用，用户对现有的 AWS 服务有了更多的功能性需求。比如最近一段时间来，AWS 服务有一系列的功能性增强，比如 Amazon CloudFront 新增了对 cookie、应用的支持，Amazon S3 新增了对 CORS 的支持等。

2) 现有 AWS 服务在更多 Region 的部署

   Amazon 在全球部署有 AWS Region，但并非每个 Region 都支持所有的 AWS，随着需求的增强与条件成熟，每个 Region 所能支持的 AWS 也会相应增加，比如亚太新加坡 Region 最近增加了对 AWS Elastic Beanstalk 的支持等。可以预见，Amazon 还有可能在未来增加全球 Region 的部署数量。

3) 现有 AWS 服务的价格调整

   AWS 不同 Region 所提供的 AWS 服务价格有所不同，不难理解，价格波动原因一部分来自于 Region 所在地的具体情况。

4) 新的 AWS 服务

   虽然 Amazon 目前已经提供了 12 个门类 33 种云计算服务，但随着云计算的深入发展和普及，这个数目肯定不会是最终的结果。比如，最近（2012 年 8 月 20 日）新上线的 Amazon Glacier 服务，提供磁带机级别的归档与备份服务，使 Amazon 的存储产品线更加充实。预期未来 Amazon 还会推出更多新的服务，AWS 在计算和存储方面的服务已经比较丰富，但在大数据处理方面，只有 Amazon Elastic MapReduce，或许 Amazon 将来会推出离线或在线数据处理的云计算服务，并提供大量数据处理基本算法库的支持。

参考文献

[1]	https://www.mturk.com
[2]	http://profitimes.com/news/amazons-next-billion-dollar-business-eyed/
[3]	http://aws.amazon.com/ec2/
[4]	https://aws.amazon.com/amis/
[5]	http://aws.amazon.com/s3/
[6]	http://aws.amazon.com/rds/
[7]	http://aws.amazon.com/simpledb/
[8]	http://aws.amazon.com/sqs/
[9]	http://aws.amazon.com/ebs/
[10]	http://aws.amazon.com/cloudwatch/
[11]	https://aws.amazon.com/vpc
[12]	http://d36cz9buwru1tt.cloudfront.net/pdf/AWS_Security_Whitepaper.pdf
[13]	https://aws.amazon.com/ec2-sla
[14]	http://aws.amazon.com/dedicated-instances
[15]	https://aws.amazon.com/ec2/purchasing-options
[16]	https://aws.amazon.com/ec2/spot-instances
[17]	http://aws.amazon.com/marketplace/ref=mkt_ste_ec2
[18]	https://aws.amazon.com/ec2/instance-types/
[19]	https://aws.amazon.com/govcloud-us/
[20]	http://aws.amazon.com/autoscaling/
[21]	https://aws.amazon.com/elasticloadbalancing
[22]	https://aws.amazon.com/hpc-applications/
[23]	https://aws.amazon.com/ec2/vmimport/
[24]	http://aws.amazon.com/marketplace/ref=mkt_ste_ec2
[25]	http://aws.amazon.com/elasticmapreduce/
[26]	http://docs.amazonwebservices.com/ElasticLoadBalancing/latest/DeveloperGuide
[27]	https://aws.amazon.com/cloudfront/
[28]	https://aws.amazon.com/route53/
[29]	https://aws.amazon.com/cloudfront/sla/
[30]	https://aws.amazon.com/console
[31]	https://d2868cy5s1ejmq.cloudfront.net/cloudfront-diagram
[32]	http://d1k5ny0m6d4zlj.cloudfront.net/diag/CFStreamingDiag.html
[33]	http://docs.amazonwebservices.com/AmazonCloudFront/latest/DeveloperGuide/LiveStreamingAdobeFMS4.5.html
[34]	http://docs.amazonwebservices.com/AmazonCloudFront/latest/DeveloperGuide/IISLiveSmoothStreaming4.1.html
[35]	http://aws.amazon.com/cloudfront-request/

[36]	https://aws.amazon.com/calculator/
[37]	http://aws.amazon.com/rds/faqs/#75
[38]	http://aws.amazon.com/rds/faqs/#118
[39]	https://aws.amazon.com/rds/mysql/#Read_Replica
[40]	http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/USER_VPC.html
[41]	https://aws.amazon.com/rds/reserved-instances/
[42]	http://aws.amazon.com/rds/mysql/#features
[43]	http://aws.amazon.com/rds/oracle/#features
[44]	http://aws.amazon.com/rds/sqlserver/#features
[45]	http://aws.amazon.com/rds/mysql/#Multi-AZ
[46]	http://aws.amazon.com/dynamodb/
[47]	http://aws.amazon.com/dynamodb/#access_model_api_overview
[48]	http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/API.html
[49]	https://aws.amazon.com/elasticache/
[50]	http://code.google.com/p/memcached/wiki/Clients
[51]	https://aws.amazon.com/elasticache/faqs#Can_I_control_software_upgrades_to_new_supported_versions
[52]	https://aws.amazon.com/sns/
[53]	https://aws.amazon.com/elasticache/faqs#How_do_I_control_access
[54]	https://aws.amazon.com/elasticache/faqs#What_are_Cache_Parameter_groups
[55]	https://aws.amazon.com/iam/
[56]	http://aws.amazon.com/code/1288653099190193
[57]	http://aws.amazon.com/mfa/
[58]	http://docs.amazonwebservices.com/IAM/latest/APIReference/
[59]	https://aws.amazon.com/storagegateway/
[60]	https://portal.aws.amazon.com/gp/aws/developer/account?ie=UTF8&action=billing-alerts
[61]	http://docs.amazonwebservices.com/AmazonCloudWatch/latest/DeveloperGuide/
[62]	https://aws.amazon.com/elasticbeanstalk/
[63]	http://aws.amazon.com/visualstudio/
[64]	http://aws.amazon.com/code/AWS-Elastic-Beanstalk/6752709412171743
[65]	https://aws.amazon.com/eclipse/
[66]	https://aws.amazon.com/cloudformation/
[67]	https://aws.amazon.com/cloudformation/aws-cloudformation-templates/
[68]	http://docs.amazonwebservices.com/AWSCloudFormation/latest/APIReference/
[69]	https://aws.amazon.com/amazon-linux-ami/
[70]	https://aws.amazon.com/cloudformation/aws-cloudformation-articles-and-tutorials/
[71]	https://aws.amazon.com/cloudsearch/
[72]	http://docs.amazonwebservices.com/cloudsearch/latest/developerguide/
[73]	https://aws.amazon.com/swf/
[74]	https://aws.amazon.com/documentation/swf/
[75]	http://docs.amazonwebservices.com/sns/latest/api/
[76]	https://aws.amazon.com/ses/
[77]	http://docs.amazonwebservices.com/ses/latest/DeveloperGuide/

[78]	https://aws.amazon.com/marketplace/ref=mkt_ste_pas
[79]	https://aws.amazon.com/route53/
[80]	https://aws.amazon.com/route53/faqs/#Supported_DNS_record_types
[81]	https://aws.amazon.com/route53/sla
[82]	http://aws.amazon.com/contact-us/vpc-request/
[83]	http://docs.amazonwebservices.com/AmazonVPC/latest/UserGuide/VPC_Appendix_Limits.html
[84]	https://aws.amazon.com/directconnect/
[85]	http://en.wikipedia.org/wiki/LOA-CFA
[86]	https://aws.amazon.com/directconnect/partners
[87]	https://aws.amazon.com/fps/
[88]	https://payments.amazon.com/sdui/sdui/business?sn=devfps/basic
[89]	https://payments.amazon.com/sdui/sdui/business?sn=devfps/advanced
[90]	https://payments.amazon.com/sdui/sdui/business?sn=devfps/marketplace
[91]	https://payments.amazon.com/sdui/sdui/business?sn=devfps/aggregated
[92]	https://payments.amazon.com/sdui/sdui/business?sn=devfps/am
[93]	https://payments.amazon.com/sdui/sdui/overview
[94]	http://docs.amazonwebservices.com/AmazonFPS/2007-01-08/FPSDeveloperGuide/
[95]	https://developer.payments-sandbox.amazon.com/landingpage/
[96]	https://payments.amazon.com/sdui/sdui/about?nodeId=6120
[97]	https://payments.amazon.com/sdui/sdui/business?sn=devpricing/fpspricing
[98]	https://payments.amazon.com/sdui/sdui/business?sn=devfaq/faq#feat_discounts
[99]	https://payments.amazon.com/sdui/sdui/business?sn=cba/o
[100]	https://payments.amazon.com/sdui/sdui/business?sn=paynow/o
[101]	https://aws.amazon.com/devpay/
[102]	https://portal.aws.amazon.com/gp/aws/application/dashboard/index.html
[103]	http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=144
[104]	https://aws.amazon.com/govcloud-us/
[105]	https://aws.amazon.com/s3-sla
[106]	https://aws.amazon.com/importexport
[107]	http://docs.amazonwebservices.com/AmazonS3/latest/gsg/
[108]	https://aws.amazon.com/sdkfornet
[109]	https://aws.amazon.com/sdkforjava
[110]	http://docs.amazonwebservices.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/s3/AmazonS3EncryptionClient.html
[111]	http://docs.amazonwebservices.com/AmazonS3/latest/dev/
[112]	https://aws.amazon.com/glacier/
[113]	http://www.youtube.com/watch?v=jQR9JpVKc2A
[114]	http://docs.amazonwebservices.com/amazonglacier/latest/dev/amazon-glacier-getting-started.html
[115]	https://aws.amazon.com/publicdatasets/
[116]	http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/AmazonEBS.html
[117]	http://awsimportexport.s3.amazonaws.com/aws-import-export-calculator.html

[118]	http://www.vmware.com/products/vsphere-hypervisor/overview.html
[119]	https://aws.amazon.com/premiumsupport/
[120]	https://aws.amazon.com/resources/
[121]	https://aws.amazon.com/faqs/
[122]	https://aws.amazon.com/forums
[123]	https://aws.amazon.com/awis/
[124]	http://en.wikipedia.org/wiki/Dmoz
[125]	http://download.alexa.com/
[126]	https://aws.amazon.com/alexatopsites/
[127]	http://www.alexa.com/help/traffic-learn-more
[128]	http://www.behind-the-enemy-lines.com/2010/12/mechanical-turk-now-with-4092-spam.html
[129]	https://d36cz9buwru1tt.cloudfront.net/AWS_Web_Hosting_Best_Practices.pdf
[130]	https://aws.amazon.com/web-mobile-social/
[131]	https://aws.amazon.com/game-hosting/
[132]	https://aws.amazon.com/articles/1044
[133]	https://aws.amazon.com/solutions/case-studies/gumi-english/
[134]	https://aws.amazon.com/media-sharing/
[135]	https://aws.amazon.com/solutions/case-studies/photowall/
[136]	https://aws.amazon.com/business-applications/
[137]	https://aws.amazon.com/solutions/case-studies/ratb/
[138]	https://aws.amazon.com/solutions/global-solution-providers/oracle/
[139]	https://aws.amazon.com/sap/
[140]	http://www.awsmicrosite.com/
[141]	https://aws.amazon.com/solutions/global-solution-providers/ibm/
[142]	https://aws.amazon.com/big-data/
[143]	http://www.netlib.org/benchmark/hpl/
[144]	http://media.amazonwebservices.com/AWS_Disaster_Recovery.pdf
[145]	https://aws.amazon.com/solutions/case-studies/whats-up-interactive/