UNIT V
EXPLORING INFRASTRUCTURE AS A SERVICE

5.1 Understanding Amazon Web Services

Amazon Web Services (AWS) is a comprehensive and widely adopted cloud computing platform provided by Amazon. It offers a broad set of global compute, storage, database, analytics, machine learning, and other services that help organizations move faster, lower IT costs, and scale applications.

Key Characteristics of AWS:

1. **Global Infrastructure:**
   - AWS operates data centers in multiple regions around the world, allowing customers to deploy applications globally with low latency and high availability.
2. **Elasticity and Scalability:**
   - AWS services are designed to scale up and down based on demand, allowing customers to quickly and easily add or remove resources as needed.
3. **Pay-as-You-Go Pricing:**
   - AWS offers a pay-as-you-go pricing model, where customers only pay for the compute power, storage, and other resources they use, with no long-term contracts or upfront commitments.
4. **Security and Compliance:**
   - AWS adheres to stringent security standards and certifications, helping customers meet security and compliance requirements for various industries and geographic regions.
5. **Wide Range of Services:**
   - AWS provides over 200 fully featured services across compute, storage, databases, networking, analytics, machine learning, artificial intelligence (AI), Internet of Things (IoT), security, and more.

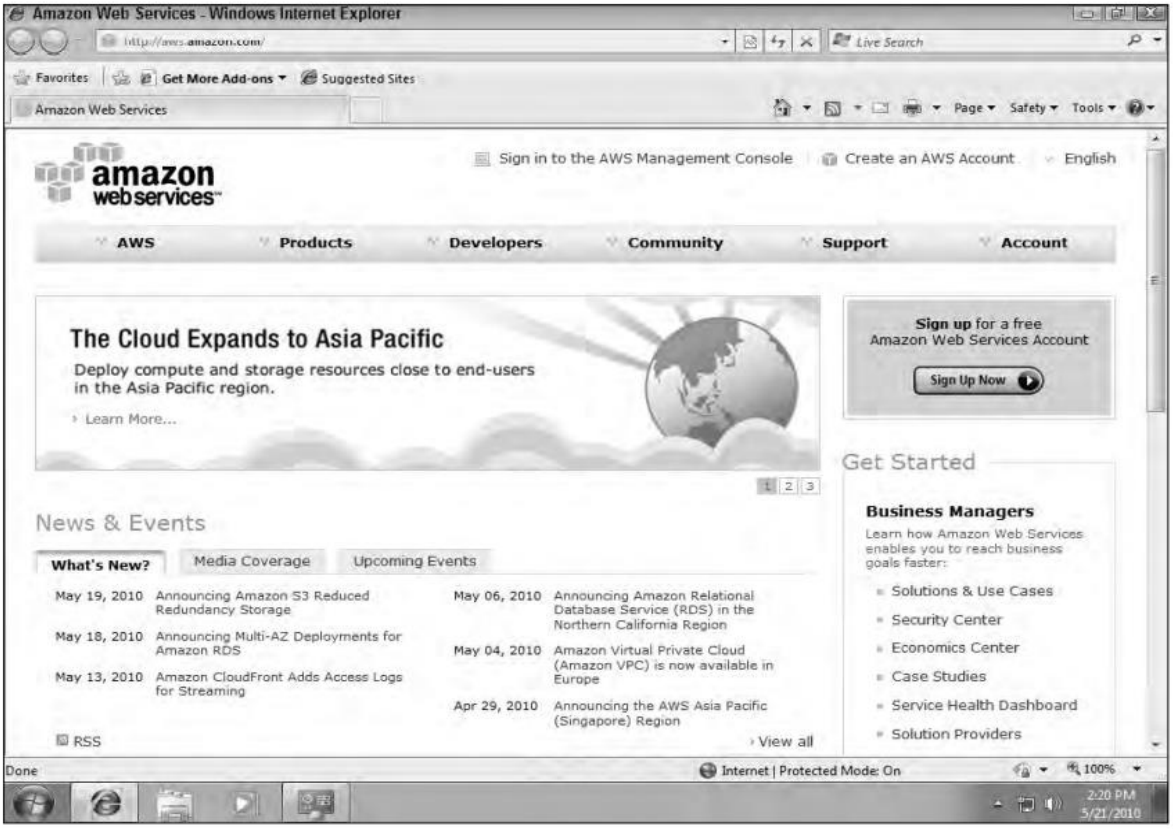


Fig 5.1 Amazon Web Services home page

5.2 Amazon Web Service Components and Services

Amazon Web Services (AWS) provides a vast array of cloud computing services and solutions to support businesses of all sizes across various industries. Here's an overview of some key AWS components and services:

Compute Services

1. **Amazon EC2 (Elastic Compute Cloud):**
   - Virtual servers (instances) with resizable compute capacity in the cloud.
   - Supports a wide range of operating systems and instance types.
2. **AWS Lambda:**
   - Serverless computing service.
   - Runs code in response to events and automatically scales.
3. **AWS Elastic Beanstalk:**
   - Platform as a Service (PaaS) for deploying and managing applications.
   - Supports multiple programming languages and frameworks.

Storage Services

1. **Amazon S3 (Simple Storage Service):**
   - Object storage service.
   - Scalable, durable, and highly available storage for a variety of use cases.
2. **Amazon EBS (Elastic Block Store):**
   - Persistent block-level storage volumes for use with EC2 instances.
   - Supports different volume types including SSD and HDD.
3. **Amazon Glacier:**
   - Low-cost cloud storage for data archival and long-term backup.
   - Retrieval times from minutes to hours.
4. **Amazon EFS (Elastic File System):**
   - Fully managed file storage service.
   - Supports NFSv4 protocol and scales automatically.

Database Services

1. **Amazon RDS (Relational Database Service):**
   - Managed relational database service.
   - Supports MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora.
2. **Amazon DynamoDB:**
   - Fully managed NoSQL database service.
   - Supports key-value and document data models with automatic scaling.
3. **Amazon Redshift:**
   - Fully managed data warehouse service.
   - Designed for analytics and querying large datasets using SQL.
4. **Amazon DocumentDB (with MongoDB compatibility):**
   - Fully managed document database service.
   - Compatible with MongoDB workloads.

Networking Services

1. **Amazon VPC (Virtual Private Cloud):**
   - Virtual network dedicated to your AWS account.
   - Provides control over network configuration (IP address range, subnets, route tables).
2. **AWS Direct Connect:**
   - Establishes a dedicated network connection from your premises to AWS.
   - Improves data transfer speed and reduces network costs.
3. **Amazon Route 53:**
   - Scalable Domain Name System (DNS) web service.
   - Routes end users to internet applications by translating names (like www.example.com) into IP addresses.

Management and Monitoring

1. **AWS CloudWatch:**
    - Monitoring and observability service.
    - Collects and tracks metrics, monitors log files, sets alarms, and automatically reacts to changes in AWS resources.
2. **AWS CloudFormation:**
    - Infrastructure as Code (IaC) service.
    - Automates provisioning and management of AWS resources using templates.
3. **AWS Systems Manager:**
    - Unified interface for managing AWS resources.
    - Automates operational tasks, maintains system compliance, and patches instances.

Security, Identity, and Compliance

1. **AWS IAM (Identity and Access Management):**
    - Manages access to AWS services and resources securely.
    - Controls who can use AWS resources (authentication) and what actions they can perform (authorization).
2. **AWS Shield:**
    - Managed Distributed Denial of Service (DDoS) protection service.
    - Protects applications running on AWS against DDoS attacks.
3. **AWS WAF (Web Application Firewall):**
    - Protects web applications from common web exploits.
    - Integrates with CloudFront and Application Load Balancer.

AI/ML Services

1. **Amazon SageMaker:**
    - Fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning models quickly.
2. **Amazon Comprehend:**
    - Natural language processing (NLP) service that uses machine learning to find insights and relationships in a text.
3. **Amazon Rekognition:**
    - Deep learning-based image and video analysis service that can identify objects, people, text, scenes, and activities.

Additional Services

1. **AWS Lambda:**
    - Serverless compute service that runs your code in response to events and automatically manages the compute resources for you, making it easy to build applications that respond quickly to new information.
2. **Amazon SQS:**
    - Fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.
3. **Amazon SNS:**
    - Fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication.
4. **AWS IoT:**
    - A managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices.

5.3 Working with the Elastic Compute Cloud (EC2)

Working with Amazon Elastic Compute Cloud (Amazon EC2) involves several key tasks to provision, manage, and interact with virtual servers in the cloud. Here's a guide on how to work with EC2:

1. Launching an EC2 Instance

To get started with EC2, you typically begin by launching an instance, which is a virtual server in the cloud.

1. **Navigate to the EC2 Dashboard:**

- Log in to the AWS Management Console.
- Go to the EC2 service dashboard by searching for "EC2" and selecting it.

2. **Launch Instance:**
   - Click on the "Instances" link in the left sidebar and then click the "Launch Instance" button.
3. **Choose an Amazon Machine Image (AMI):**
   - Select an AMI that suits your needs, such as Amazon Linux, Ubuntu, Windows Server, etc.
   - You can also select AWS Marketplace AMIs or your own custom AMIs.
4. **Choose an Instance Type:**
   - Select the instance type based on your application requirements (e.g., t2.micro for basic testing, m5.large for more resources).
   - Each instance type offers different combinations of CPU, memory, storage, and networking capacity.
5. **Configure Instance Details:**
   - Configure additional settings like network, subnet, IAM role, and instance shutdown behavior if needed.
   - Advanced users may configure options like user data scripts here for instance initialization.
6. **Add Storage:**
   - Specify the size and type (e.g., General Purpose SSD) of the root EBS volume.
   - You can add additional volumes if needed for data storage.
7. **Add Tags:**
   - Optionally, add tags to your instance for better organization and management.
8. **Configure Security Group:**
   - Create a new security group or select an existing one.
   - Configure inbound and outbound rules to control traffic to your instance (e.g., SSH, HTTP, HTTPS).
9. **Review and Launch:**
   - Review your instance configuration.
   - Click "Launch" to start the instance.
10. **Select or Create a Key Pair:**
    - Choose an existing key pair or create a new one.
    - This key pair allows you to securely SSH into your instance.
11. **Access Your Instance:**
    - Once the instance is launched, note its Public DNS or IP address from the EC2 dashboard.

2. Connecting to Your EC2 Instance

After launching your EC2 instance, you can connect to it using SSH (for Linux/Mac) or PuTTY (for Windows).

- **SSH Connection (Linux/Mac):**

bash
Copy code
```
ssh -i /path/to/your-key.pem ec2-user@public-dns-or-ip
```

- **Using PuTTY (Windows):**
  - Convert your .pem key file to a .ppk file using PuTTYgen.
  - Use PuTTY to connect to your instance:
    - Enter your instance's Public IP address or DNS name under "Session."
    - Load your .ppk private key under "Connection > SSH > Auth."
    - Click "Open" to connect.

3. Managing Your EC2 Instance

Once connected to your EC2 instance, you can perform various management tasks:

- **Installing Software:**
  - Use package managers (yum for Amazon Linux, apt for Ubuntu) to

install software packages.

- **Configuring Security:**
  - Update firewall rules (security groups) using the AWS Management Console or AWS CLI.
- **Monitoring and Logging:**
  - Use Amazon CloudWatch to monitor EC2 instances and set alarms for metrics like CPU utilization, network traffic, etc.
- **Scaling and Load Balancing:**
  - Use Auto Scaling groups to automatically scale your EC2 fleet based on demand.
  - Implement Elastic Load Balancing to distribute incoming traffic across multiple instances.
- **Backup and Restore:**
  - Create snapshots of your EBS volumes for backups and restore them when needed.
- **Instance Maintenance:**
  - Regularly apply OS patches and updates to keep your instances secure.

4. Terminating an Instance

When you no longer need an instance, terminate it to avoid unnecessary charges:

- Go to the EC2 dashboard, select the instance, and click "Actions > Instance State > Terminate."
- Confirm the termination. Note that terminating an instance deletes all data on its EBS volumes unless you create snapshots beforehand.

Type	Compute Engine	RAM (GB)	Storage (GB)1	Platform	I/O Performance	API Name
Micro instance	*Up to 2 EC2 Compute Units (1 virtual core) in short bursts*	*0.613*	*EBS (Elastic Block Storage) storage only*	32-bit or 64-bit	Low	T1.micro
Standard instance – small (default)	1 EC2 Compute Unit (1 virtual core)	1.7	160	32-bit	Moderate	m1.small
Standard instance – large	4 EC2 Compute Units (2 virtual cores X 2 EC2 Units)	7.5	850	64-bit	High	m1.large
Standard instance – extra large	8 EC2 Compute Units (4 virtual cores X 2 EC2 Units)	15	1,690	64-bit	High	m1.xlarge
High Memory Double Extra Large Instance	13 EC2 Compute Units (4 virtual cores X 3.25 EC2 Units)	34.2	850	64-bit	High	m2.2xlarge

High Memory Quadruple Extra Large Instance	26 EC2 Compute Units (8 virtual cores X 3.25 EC2 Units)	68.4	1,690	64-bit	High	m2.4xlarge
High CPU Medium Instance	5 EC2 Compute Units (2 virtual cores X 2.5 EC2 Units)	1.7	350	32-bit	Moderate	c1.medium
High CPU Extra Large Instance	20 EC2 Compute Units (8 virtual cores X 2.5 EC2 Units)	7	1,690	64-bit	High	c1.xlarge

1. Storage is not persistent. All assigned storage is lost upon rebooting. To store data on AWS, you need to create a Simple Storage Service (S3) bucket or an Elastic Block Storage (EBS) volume.

Table 5.1 Amazon Machine Image Instance Types

5.3.2 Pricing models The pricing of these different AMI types depends on the operating system used, which data center the AMI is located in (you can select its location), and the amount of time that the AMI runs. Rates are quoted based on an hourly rate.

Additional charges are applied for:

- the amount of data transferred
- whether Elastic IP Addresses are assigned
- your virtual private server's use of Amazon Elastic Block Storage (EBS)
- whether you use Elastic Load Balancing for two or more servers

Other features AMIs that have been saved and shut down incur a small one-time fee, but do not incur additional hourly fees.

The three different pricing models for EC2 AMIs are as follows:

On-Demand Instance: This is the hourly rate with no long-term commitment.

Reserved Instances: This is a purchase of a contract for each instance you use with a significantly lower hourly usage charge after you have paid for the reservation.

Spot Instance: This is a method for bidding on unused EC2 capacity based on the current spot price. This feature offers a significantly lower price, but it varies over time or may not be available when there is no excess capacity.

5.3.3 System images and software You can choose to use a template AMI system image with the operating system of your choice or create your own system image that contains your custom applications, code libraries, settings, and data. Security can be set through passwords, Kerberos tickets, or certificates. These operating systems are offered:

- Red Hat Enterprise Linux
- OpenSuse Linux
- Ubuntu Linux
- Sun OpenSolaris Fedora
- Gentoo Linux
- Oracle Enterprise Linux
- Windows Server 2003/2008 32-bit and 64-bit up to Data Center Edition
- Debian

Table 5.2 lists some of the more common enterprise applications that are available from AWS either as part of its canned templates or for use in building your own AMI system image. Hundreds of free and paid AMIs can be found on AWS.

Application Type	Software
Application Development Environments	IBM sMash, JBoss Enterprise Application Platform, and Ruby on Rails
Application Servers	IBM WebSphere Application Server, Java Application Server, and Oracle WebLogic Server
Batch Processing	Condor, Hadoop, and Open MPI
Databases	IBM DB2, IBM Informix Dynamic Server, Microsoft SQL Server Standard 2005, MySQL Enterprise, and Oracle Database 11g
Video Encoding and Streaming	Windows Media Server and Wowza Media Server Pro
Web Hosting	Apache HTTP, IIS/ASP.Net, IBM Lotus Web Content Management, and IBM WebSphere Portal Server

Table 5.2 EC2 Enterprise Software Types

5.3.4 Creating an account and instance on EC2

Creating an AWS Account

1. **Sign Up for an AWS Account:**
   - Go to the AWS Signup Page and click "Create an AWS Account."
   - Follow the instructions to complete the signup process, providing your email address, password, and account name.
   - Enter your contact information, payment details, and verify your identity via a phone call or text message.
   - Choose a support plan (Basic Support is free).
2. **Log In to the AWS Management Console:**
   - Once your account is created, go to the AWS Management Console and log in using your new account credentials.

Launching an Instance on Amazon EC2

1. **Navigate to the EC2 Dashboard:**
   - In the AWS Management Console, search for "EC2" and select "EC2" from the services list.
2. **Launch an Instance:**
   - Click the "Launch Instance" button.
3. **Choose an Amazon Machine Image (AMI):**
   - Select an AMI that best suits your needs. You can choose from a variety of operating systems such as Amazon Linux, Ubuntu, Windows, etc.
   - For beginners, the "Amazon Linux 2 AMI" is a good choice as it's free-tier eligible.
4. **Choose an Instance Type:**
   - Select the instance type based on your workload requirements. For most basic tasks, the "t2.micro" instance type (which is free-tier eligible) is sufficient.
   - Click "Next: Configure Instance Details."
5. **Configure Instance Details:**
   - Configure the instance settings as per your requirements. For a basic setup, the default settings are usually sufficient.
   - Click "Next: Add Storage."
6. **Add Storage:**

- Modify the storage size if needed. By default, the AMI comes with a preset storage size. You can increase it based on your application needs.
- Click "Next: Add Tags."

7. **Add Tags:**
   - (Optional) Add tags to your instance for better organization. For example, you can add a tag with a key of "Name" and a value of "MyFirstInstance."
   - Click "Next: Configure Security Group."
8. **Configure Security Group:**
   - Create a new security group or select an existing one. A security group acts as a virtual firewall for your instance.
   - Add rules to allow specific types of traffic to your instance. For example, add a rule to allow SSH traffic (port 22) from your IP address.
   - Click "Review and Launch."
9. **Review and Launch:**
   - Review your instance configuration and click "Launch."
   - You will be prompted to select an existing key pair or create a new one. A key pair is used for SSH access to your instance.
   - If you don't have a key pair, select "Create a new key pair," give it a name, and download the key pair file (.pem). Store it securely as you will need it to access your instance.
   - Click "Launch Instances."
10. **Access Your Instance:**
    - After your instance is launched, go back to the EC2 dashboard and click "Instances" to see your running instance.
    - Note the public IP address or DNS name of your instance.

Connecting to Your EC2 Instance

1. **Open a Terminal or Command Prompt:**
   - On your local machine, open a terminal (Linux/Mac) or command prompt (Windows).
2. **Change Permissions of the Key Pair File:**
   - Run the following command to ensure your key pair file has the correct permissions.

shell
Copy code
chmod 400 /path/to/your-key-pair-file.pem

3. **Connect to Your Instance:**
   - Use the SSH command to connect to your instance. Replace ec2-user with the appropriate username for your AMI, and replace the public-ip with your instance's public IP address.

```
shell
Copy code
ssh -i /path/to/your-key-pair-file.pem ec2-user@public-ip
```

For example:

```
shell
Copy code
ssh -i /path/to/my-key-pair.pem ec2-user@54.123.45.67
```

- If you are using a Windows machine and need an SSH client, you can use PuTTY. Convert the `.pem` file to a `.ppk` file using PuTTYgen and use PuTTY to connect.

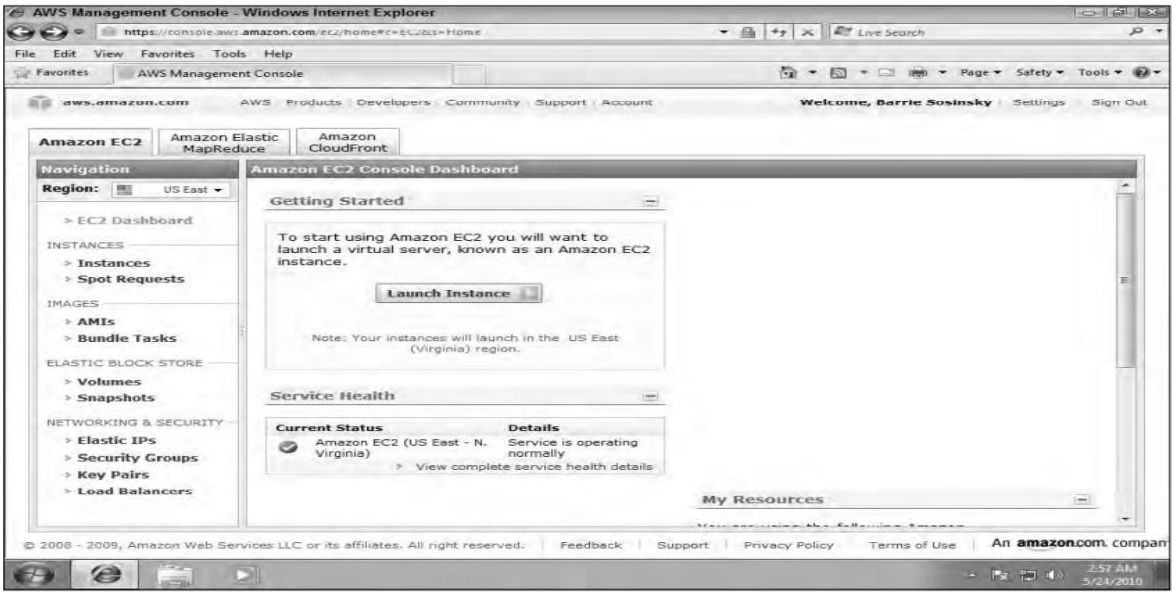


Fig 5.3 The AWS EC2 Management Console with no instances

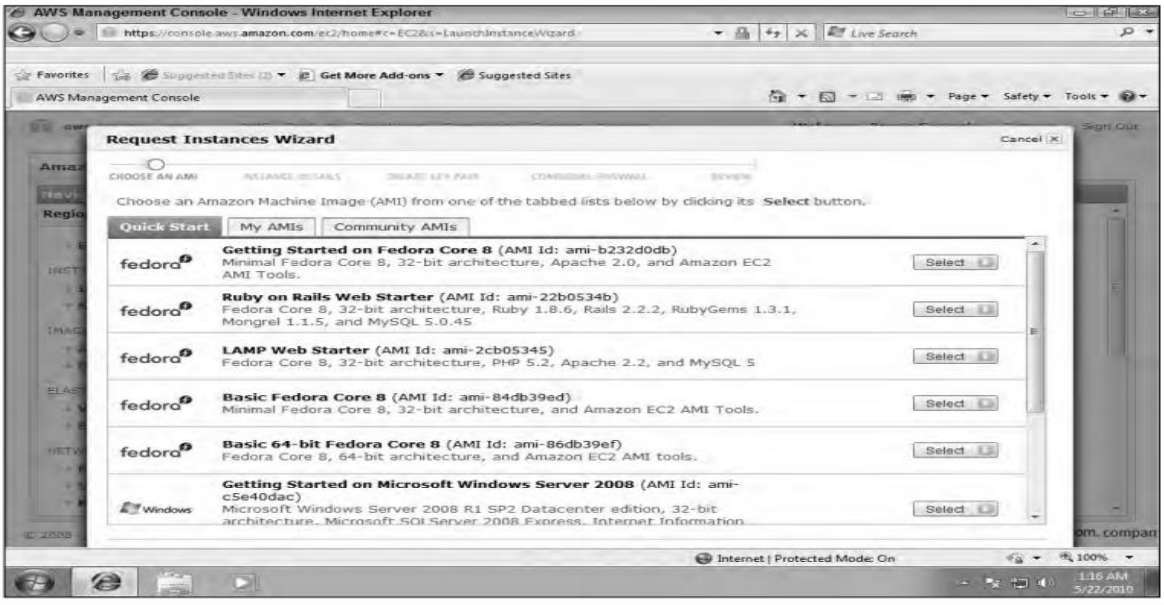


Fig 5.4 Select an Instance type from one of the templates shown, or create your own AMI in this step.

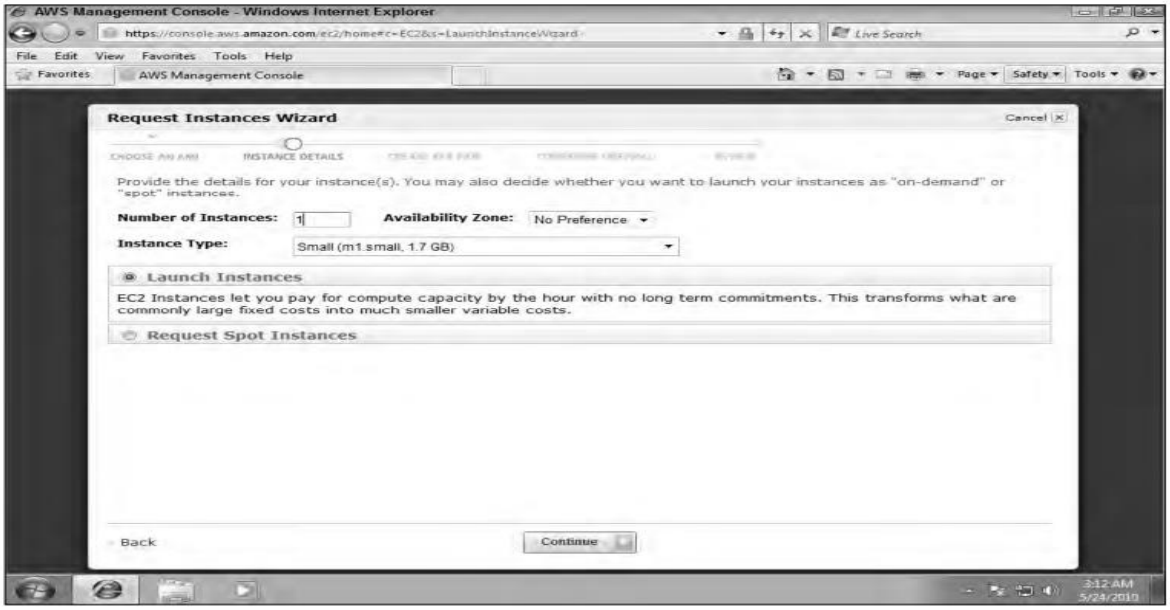
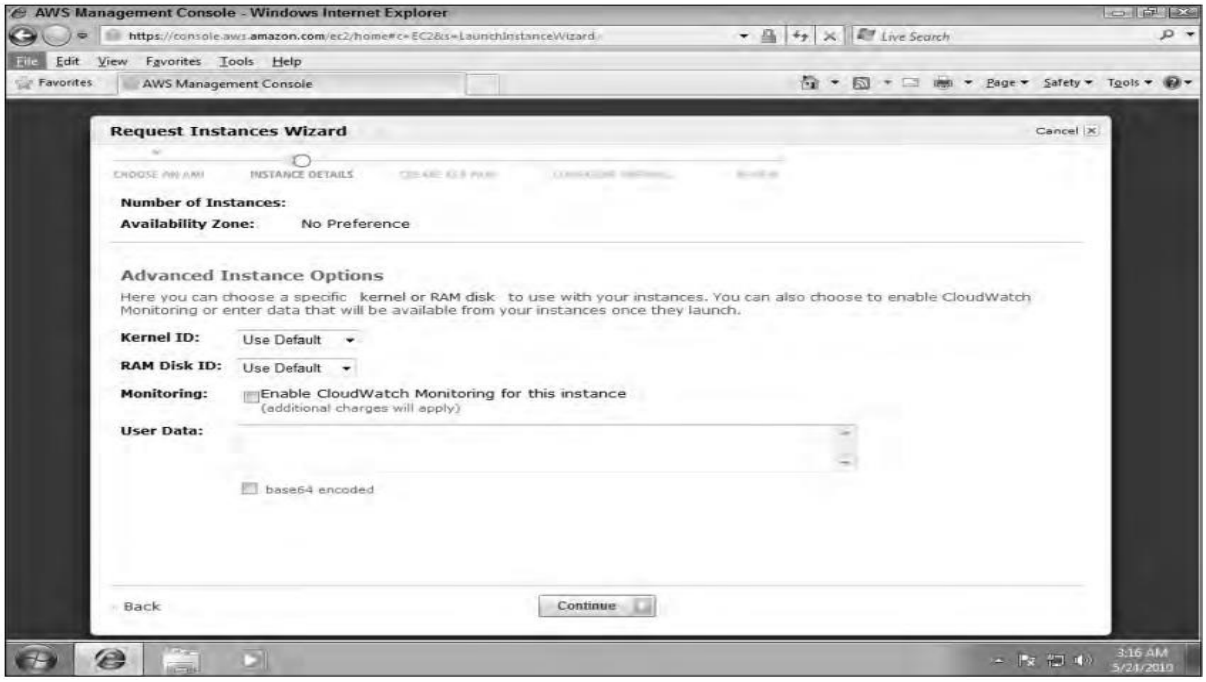


Fig 5.5 Fill in the instance details in this step.



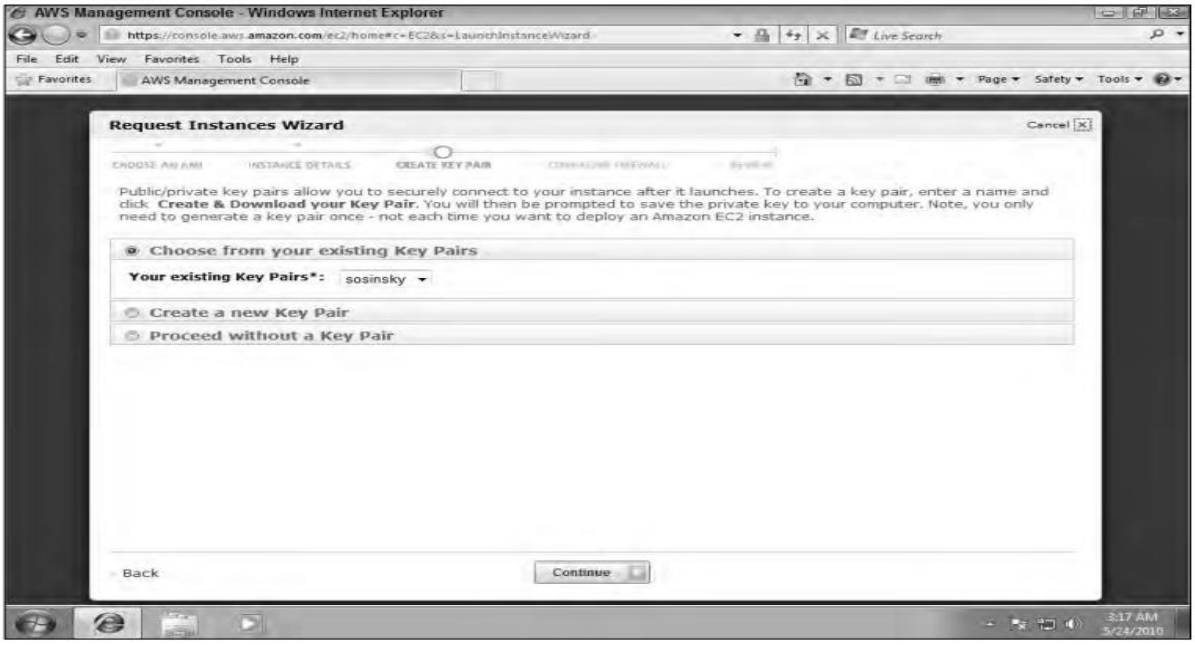
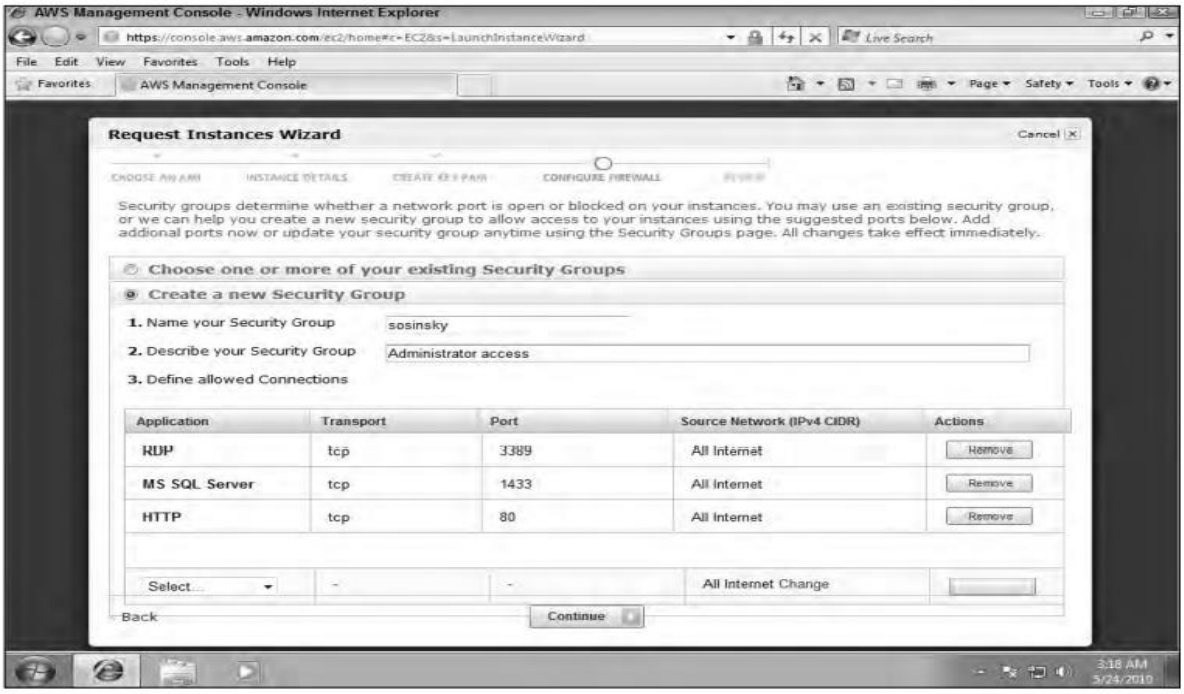


FIGURE 5.8 Firewall settings allow you to filter by service and protocol, as well as set a security group membership for access



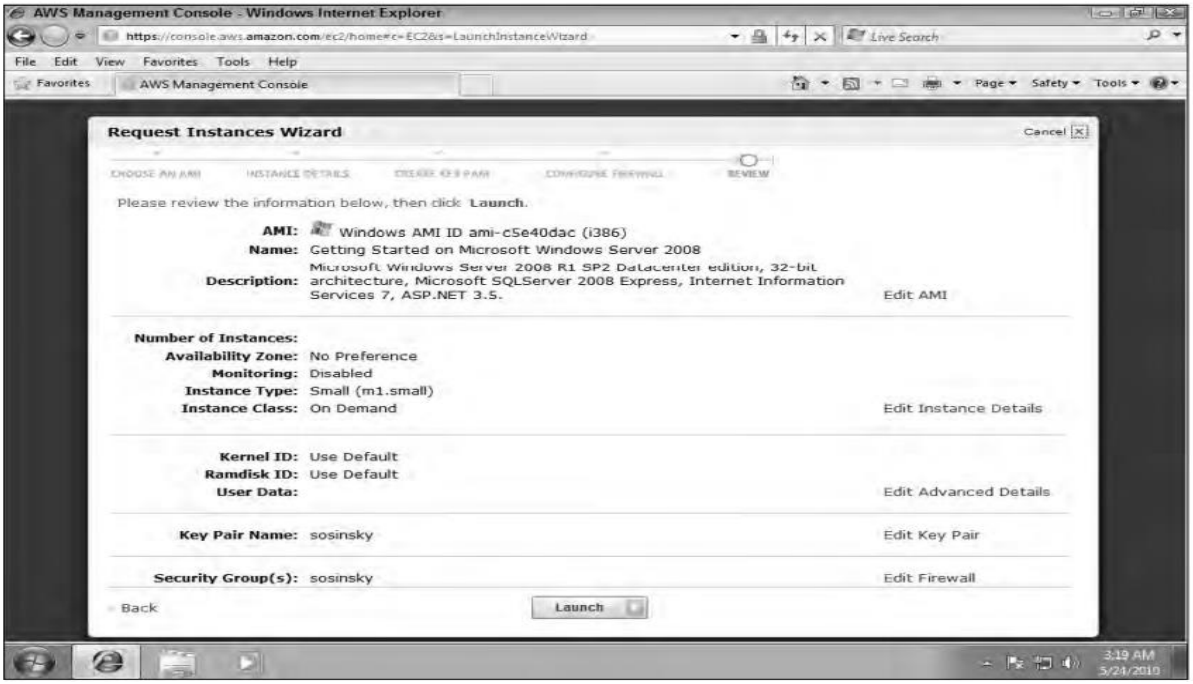


FIGURE 5.10 The AWS Management Console with an active AMI showing

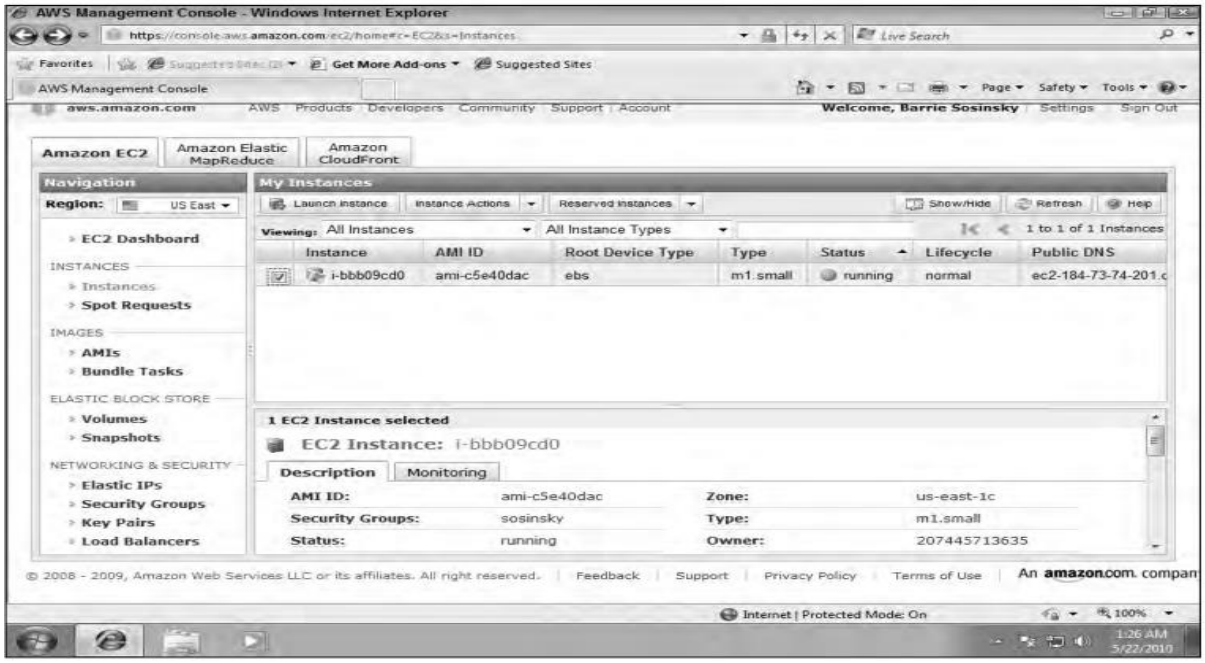
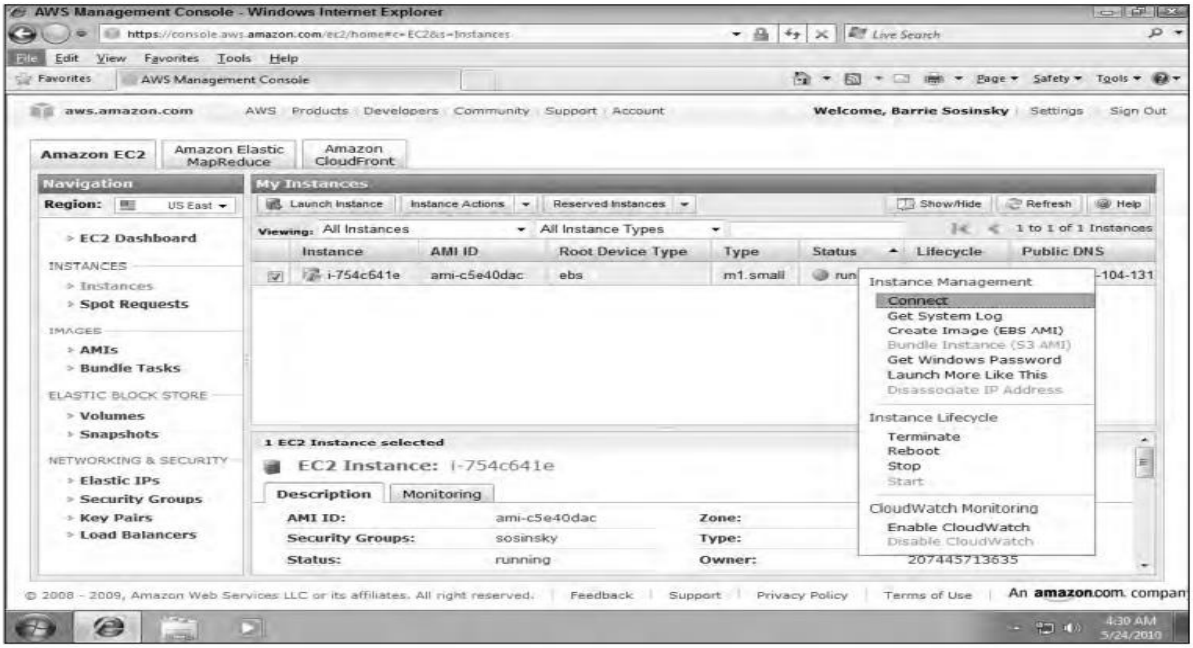


FIGURE 5.11 Context menu for a Windows system image running in an AMI



3.4 Amazon Simple Storage System (S3)

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This service is designed to store and retrieve any amount of data from anywhere on the web. It is widely used for various use cases such as backup and restore, archive, big data analytics, disaster recovery, cloud-native applications, and content storage and distribution.

Key Features of Amazon S3

1. **Scalability and Performance:**
   - Amazon S3 automatically scales to handle large volumes of data and high request rates, providing consistent and low-latency performance.
2. **Durability and Availability:**
   - S3 is designed for 99.999999999% (11 9's) durability by redundantly storing objects across multiple devices and facilities.
   - High availability with an SLA-backed uptime of 99.9% for the Standard storage class.
3. **Storage Classes:**
   - **S3 Standard:** General-purpose storage for frequently accessed data.
   - **S3 Intelligent-Tiering:** Automatically moves data between two access tiers (frequent and infrequent) to optimize costs.
   - **S3 Standard-IA (Infrequent Access):** Lower-cost option for infrequently accessed data with rapid access when needed.
   - **S3 One Zone-IA:** Lower-cost option for infrequently accessed data that is stored in a single Availability Zone.
   - **S3 Glacier:** Low-cost storage for archival data with retrieval times ranging from minutes to hours.
   - **S3 Glacier Deep Archive:** Lowest-cost storage for data that is rarely accessed, with retrieval times of up to 12 hours.
4. **Security and Compliance:**

- **Data Encryption:** Supports encryption at rest and in transit (SSL/TLS). Server-side encryption (SSE) with S3 managed keys (SSE-S3), AWS Key Management Service (SSE-KMS), or customer-provided keys (SSE-C).
- **Access Controls:** Fine-grained access control using AWS Identity and Access Management (IAM) policies, bucket policies, and Access Control Lists (ACLs).
- **Compliance:** Designed to meet various compliance programs, including PCI-DSS, HIPAA/HITECH, FedRAMP, EU Data Protection Directive, and FISMA.

5. **Storage Management:**
   - **Lifecycle Policies:** Define rules to automatically transition objects between storage classes and expire objects after a specified period.
   - **Object Lock:** Prevents object deletion or modification for a specified retention period to meet regulatory and compliance requirements.
   - **Versioning:** Keeps multiple versions of an object to recover from unintended user actions and application failures.
6. **Data Transfer and Access:**
   - **Amazon S3 Transfer Acceleration:** Speeds up data transfers to and from S3 using Amazon CloudFront's globally distributed edge locations.
   - **Multipart Upload:** Supports uploading large objects in parts to improve upload efficiency and resiliency.
   - **S3 Select:** Retrieves only a subset of data from an object using SQL expressions, reducing the amount of data transferred and accelerating query performance.
7. **Integration with Other AWS Services:**
   - **Compute:** Direct integration with Amazon EC2, AWS Lambda, and Amazon Elastic Kubernetes Service (EKS).
   - **Analytics:** Integration with Amazon Redshift, Amazon Athena, and AWS Glue for big data analytics.
   - **Machine Learning:** Direct access for training data in services like Amazon SageMaker.
   - **Data Transfer:** Integration with AWS Snowball and AWS DataSync for large-scale data migration.

5.4.2 Amazon Elastic Block Store (EBS)

Amazon Elastic Block Store (EBS) is a scalable, high-performance block storage service provided by Amazon Web Services (AWS). It is designed to work with Amazon EC2 (Elastic Compute Cloud) instances and provides persistent storage that can be attached to these instances. EBS is ideal for a wide range of applications, including databases, file systems, and enterprise applications that require high availability and consistent performance.

Key Features of Amazon EBS

1. **Persistent Storage:**
   - EBS volumes are persistent, meaning data is retained even after an EC2 instance is stopped or terminated.
2. **Scalability:**
   - EBS volumes can be resized and modified while in use, allowing for easy scalability to meet changing storage requirements.
3. **Performance:**
   - EBS offers various volume types with different performance characteristics to meet diverse application needs, including SSD-backed volumes for high-performance workloads and HDD-backed volumes for throughput-intensive applications.
4. **Snapshots:**
   - EBS supports point-in-time snapshots of volumes, which can be used for backups, data migration, and disaster recovery. Snapshots are stored in Amazon S3 and can be used to create new volumes.
5. **Encryption:**
   - EBS provides encryption at rest and in transit, using AWS Key Management Service (KMS) to manage encryption keys. This ensures data security and compliance with regulatory requirements.
6. **Availability and Durability:**
   - EBS volumes are designed for high availability and durability. Data is automatically replicated within the same Availability Zone to protect against hardware failures.
7. **Elastic Volumes:**
   - EBS allows for dynamic modification of volume size, performance characteristics, and volume type without downtime, providing flexibility to adapt to workload changes.

EBS Volume Types

1. **General Purpose SSD (gp2 and gp3):**
   - Balanced price and performance for a wide variety of workloads.
   - gp3 offers baseline performance and the ability to provision additional IOPS and throughput independently.
2. **Provisioned IOPS SSD (io1 and io2):**
   - Designed for I/O-intensive applications such as databases.
   - Offers high and consistent IOPS and low latency.
3. **Throughput Optimized HDD (st1):**
   - Low-cost HDD volume for frequently accessed, throughput-intensive workloads such as big data and log processing.
4. **Cold HDD (sc1):**
   - Lowest-cost HDD volume for infrequently accessed data with lower throughput requirements.

Property	AMI Instance	Amazon Simple Storage Service (S3)	Amazon Elastic Block Storage (EBS)	Amazon CloudFront
Adaptability	Medium	Low	High	Medium
Best usage	Transient data storage	Persistent or archival storage	Operational data storage	Data sharing and large data object streaming
Cost	Low	Medium	High	Low
Ease of use	Low	High	High	High
Data protection	Very Low	Very High	High	Low
Latency	Medium	Low	High	High
Least best used as	Persistent storage	Operational storage	For small I/O transfers	Operational data
Reliability	High	Medium	High	Medium
Throughput	Variable	Slow	High	High

TABLE 9.3: EC2 Storage Type Properties

5.5 Understanding Amazon Database Services

Amazon offers two different types of database services:

5.5.1 Amazon SimpleDB, Amazon SimpleDB is a highly available, scalable, and flexible non-relational data store that allows users to store and query structured data with minimal administrative burden. SimpleDB is designed for applications that require simplicity and scalability, providing a straightforward way to store and retrieve data without the complexities of managing a relational database.

Key Features of Amazon SimpleDB

1. **Schema-Free Data Model:**
   - SimpleDB uses a schema-free model, meaning there are no predefined columns or data types. Users can store varying attributes for each item, allowing for flexible data structures.
2. **High Availability:**
   - SimpleDB automatically replicates data across multiple data centers in an AWS region, ensuring high availability and durability.
3. **Scalability:**
   - SimpleDB scales automatically to handle increased workloads. There are no limits to the amount of data that can be stored, and the service can handle large numbers of concurrent queries.
4. **Flexible Querying:**
   - SimpleDB provides a simple query interface to retrieve data. Users can perform searches using multiple conditions, sort results, and select specific attributes to return.
5. **Automatic Indexing:**
   - All attributes are automatically indexed, allowing for fast and efficient querying without the need for manual index management.
6. **Integrated with AWS:**
   - SimpleDB integrates seamlessly with other AWS services, such as Amazon EC2, Amazon S3, and Amazon SQS, providing a cohesive ecosystem for application development.
7. **Pay-As-You-Go Pricing:**
   - Users pay only for the resources they consume, including data storage, read and write operations, and data transfer. This makes it cost-effective for applications with varying workloads.

Use Cases for Amazon SimpleDB

1. **Web and Mobile Applications:**
    - Ideal for applications that require a flexible data model and can benefit from the scalability and high availability of SimpleDB.
    - Examples: user profiles, session storage, and product catalogs.
2. **Logging and Monitoring:**
    - SimpleDB can be used to store and query log data, event records, and monitoring metrics, providing a simple way to analyze and report on operational data.
3. **Data Caching:**
    - SimpleDB can act as a caching layer for frequently accessed data, reducing the load on primary data sources and improving application performance.
4. **Lightweight Data Stores:**
    - Applications that require a lightweight and easy-to-use data store for temporary or semi-structured data can leverage SimpleDB's simplicity and flexibility.

Limitations of Amazon SimpleDB

While Amazon SimpleDB offers many benefits, it also has some limitations:

1. **Data Size:**
    - Each item in SimpleDB is limited to 256 attributes, and the total size of all attributes in an item cannot exceed 1 KB.
2. **Query Complexity:**
    - SimpleDB's query capabilities are less advanced compared to other database services like Amazon RDS or Amazon DynamoDB. It is suitable for simple queries but may not be ideal for complex querying and analytics.
3. **Throughput:**
    - SimpleDB is designed for small to medium-scale applications. For high-throughput and large-scale applications, Amazon DynamoDB might be a better choice.

Example Usage of Amazon SimpleDB

Here's an example workflow of using Amazon SimpleDB:

1. **Create a Domain:**
    - A domain is a collection of items, similar to a table in a relational database. Each domain can store up to 10 GB of data.

shell
Copy code
```
aws simpledb create-domain --domain-name MyDomain
```

2. **Store Data:**
    - Items are added to the domain with attributes. Each item can have different attributes.

shell
Copy code
```
aws simpledb put-attributes --domain-name MyDomain --item-name Item1 --attributes Name=John Age=30
```

3. **Query Data:**
    - Retrieve items based on specific conditions.

shell
Copy code
```
aws simpledb select --select-expression "select * from MyDomain where Age > '25'"
```

4. **Update Data:**
    - Update attributes of an existing item.

shell
Copy code
```
aws simpledb put-attributes --domain-name MyDomain --item-name Item1 --attributes Age=31
```

5. **Delete Data:**

- Delete items from the domain.

shell
Copy code
```
aws simpledb delete-attributes --domain-name MyDomain --item-name Item1
```

Integration with Other AWS Services

Amazon SimpleDB can be integrated with other AWS services to build comprehensive solutions:

- **Amazon EC2:** Store instance metadata or user session data.
- **Amazon S3:** Use SimpleDB to store metadata for objects stored in S3.
- **Amazon SQS:** Use SimpleDB for durable storage of messages processed by SQS.

Diagram of Amazon SimpleDB Architecture

lua
Copy code
```
+---------------------------+
| Client Applications |
| |
| +----------------------+ |
| | Queries | |
| +----------------------+ |
| |
+-------------+-------------+
 |
 v
+---------------------------+
| Amazon SimpleDB |
| |
| +----------------------+ |
| | Domain 1 | |
| +----------------------+ |
| | Domain 2 | |
| +----------------------+ |
| |
+-------------+-------------+
 |
 v
+---------------------------+
| Multiple AZs for |
| High Availability |
+---------------------------+
```

Conclusion

5.5.2 Amazon Relational Database Service (RDS)

Overview:

- Amazon RDS simplifies setting up, operating, and scaling relational databases in the cloud.
- Supports multiple database engines including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server.

Features:

- Automated backups, patching, and replication.
- Multi-AZ (Availability Zone) deployments for high availability.
- Read replicas for scalability.
- Automated scaling and storage resizing.

Use Cases:

- Web and mobile applications.
- E-commerce platforms.
- Business applications like ERP and CRM systems.

Type1	Compute Engine	RAM (GB)	Platform	Price2
Small DB Instance (default)	1 EC2 Compute Unit (1 virtual core)	1.7	64-bit	$0.11
Large DB Instance	2 EC2 Compute Units (2 virtual cores X 2 EC2 Units)	7.5	64-bit	$0.44
Extra Large DB Instance	8 EC2 Compute Units (4 virtual cores X 2 EC2 Units)	15	64-bit	$0.88
Double Extra Large DB Instance	13 EC2 Compute Units (4 virtual cores X 3.25 EC2 Units)	34	64-bit	$1.55
Quadruple Extra Large DB Instance	26 EC2 Compute Units (8 virtual cores X 3.25 EC2 Units)	68	64-bit	$3.10

1. Storage available is from 5GB to 1TB.

2. Price for U.S. N. Virginia deployment for database machine; storage price is $0.10 per GB-month; and I/O rate price is $0.10 per 1 million requests for the same location. Data transfer rates also apply.

Table 5.4 Amazon Relational Database Service Instance Class

5.5.3 Choosing a database for AWS

Choosing the right database service on AWS depends on various factors such as the type of data, workload characteristics, scalability requirements, query complexity, and specific application needs. Here is a guide to help you select the appropriate AWS database service based on different criteria:

Criteria for Choosing an AWS Database Service

1. **Data Type and Structure:**
   - **Relational Data:** If your data is structured, relational, and requires ACID (Atomicity, Consistency, Isolation, Durability) properties, consider Amazon RDS or Amazon Aurora.
   - **NoSQL Data:** For schema-less, unstructured, or semi-structured data, consider Amazon DynamoDB, Amazon DocumentDB, or Amazon Neptune.
   - **Time Series Data:** If you need to store and query time series data, Amazon Timestream is designed for this purpose.
   - **Ledger Data:** For applications that require a transparent, immutable, and cryptographically verifiable transaction log, use Amazon QLDB.
2. **Workload Characteristics:**
   - **High Read/Write Throughput:** If your application demands high read/write throughput with low latency, Amazon DynamoDB is a suitable choice.
   - **Data Warehousing and Analytics:** For complex queries, aggregations, and analytics on large datasets, use Amazon Redshift.
   - **In-Memory Caching:** To accelerate application performance by caching frequently accessed data, consider Amazon ElastiCache (Redis or Memcached).
3. **Scalability Requirements:**
   - **Automatic Scaling:** If automatic scaling of read/write capacity is crucial, Amazon DynamoDB and Amazon Aurora provide this feature.
   - **Manual Scaling:** If you prefer to manually control scaling, Amazon RDS offers this flexibility.
4. **Query Complexity:**
   - **Simple Queries:** For simple key-value queries, Amazon DynamoDB is ideal.
   - **Complex Queries:** For complex relational queries, joins, and transactions, Amazon RDS or Amazon Aurora are more suitable.
   - **Graph Queries:** If your application needs to traverse and query graph data, Amazon Neptune is designed for this.

5. **High Availability and Durability:**
    - **Multi-AZ Deployments:** For high availability and automated failover, consider Amazon RDS, Amazon Aurora, or Amazon DynamoDB (with global tables).
    - **Global Replication:** For applications that require data replication across multiple regions, Amazon DynamoDB and Amazon Aurora Global Database provide this capability.

AWS Database Services and Their Use Cases

1. **Amazon RDS (Relational Database Service):**
    - **Use Cases:** Web and mobile applications, e-commerce platforms, business applications (ERP, CRM).
    - **Supported Engines:** Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, SQL Server.
    - **Key Features:** Automated backups, Multi-AZ deployments, Read replicas, Automated scaling.
2. **Amazon Aurora:**
    - **Use Cases:** Enterprise applications, SaaS applications, high-performance web applications.
    - **Compatibility:** MySQL, PostgreSQL.
    - **Key Features:** Up to 5x faster than MySQL and 3x faster than PostgreSQL, Distributed storage, Global databases, Automated failover.
3. **Amazon DynamoDB:**
    - **Use Cases:** Real-time bidding, gaming, IoT applications, mobile and web apps.
    - **Data Model:** Key-value and document store.
    - **Key Features:** Fully managed, Automatic scaling, Global tables, Built-in security and backup, In-memory caching with DynamoDB Accelerator (DAX).
4. **Amazon Redshift:**
    - **Use Cases:** Data warehousing, Business intelligence, ETL operations.
    - **Data Model:** Columnar storage.
    - **Key Features:** Massively parallel processing (MPP), Redshift Spectrum (querying data in S3), Automated backups, Scaling.
5. **Amazon ElastiCache:**
    - **Use Cases:** Caching, real-time analytics, session stores, leaderboards.
    - **Supported Engines:** Redis, Memcached.
    - **Key Features:** Sub-millisecond latency, Clustering, Automated failover and backups, Integration with other AWS services.
6. **Amazon Neptune:**
    - **Use Cases:** Social networking, fraud detection, recommendation engines.
    - **Data Model:** Graph (Property graph and RDF).
    - **Key Features:** High-performance graph queries, Automated backups and patching, Multi-AZ deployments.
7. **Amazon DocumentDB (with MongoDB compatibility):**
    - **Use Cases:** Content management systems, catalogs, user profiles, mobile applications.
    - **Data Model:** Document store.
    - **Key Features:** MongoDB compatibility, Automated scaling, Multi-AZ deployments, Backup and restore.
8. **Amazon Timestream:**
    - **Use Cases:** IoT applications, DevOps monitoring, industrial telemetry.
    - **Data Model:** Time series.
    - **Key Features:** Fast ingestion, Storage tiering, SQL-based queries, Built-in data retention policies.
9. **Amazon QLDB (Quantum Ledger Database):**
    - **Use Cases:** Financial transaction tracking, supply chain management,

identity and access management.
    - **Data Model:** Ledger.
    - **Key Features:** Immutable and append-only journal, Cryptographic verification, SQL-like query capabilities with PartiQL.

Decision Tree for Choosing an AWS Database Service

1. **Is your data structured and relational?**
    - **Yes:** Consider Amazon RDS or Amazon Aurora.
    - **No:** Proceed to the next question.
2. **Is your data unstructured or semi-structured?**
    - **Yes:** Consider Amazon DynamoDB or Amazon DocumentDB.
    - **No:** Proceed to the next question.
3. **Do you need to analyze large datasets with complex queries?**
    - **Yes:** Consider Amazon Redshift.
    - **No:** Proceed to the next question.
4. **Do you need in-memory caching for low-latency access?**
    - **Yes:** Consider Amazon ElastiCache.
    - **No:** Proceed to the next question.
5. **Do you need to store and query graph data?**
    - **Yes:** Consider Amazon Neptune.
    - **No:** Proceed to the next question.
6. **Do you need to manage time series data?**
    - **Yes:** Consider Amazon Timestream.
    - **No:** Proceed to the next question.
7. **Do you need a ledger with an immutable transaction log?**
    - **Yes:** Consider Amazon QLDB.
    - **No:** Re-evaluate your specific requirements.