DNS (Amazon Route 53)
1. Amazon Route 53 is AWS's scalable DNS service. It translates domain names (e.g., www.example.com) into IP addresses and routes traffic to AWS resources (e.g., EC2, S3, ALB). It facilitates domain resolution by managing DNS records in hosted zones and applying routing policies to direct traffic efficiently.

2. Public Hosted Zones manage DNS records for publicly accessible resources (e.g., websites). Private Hosted Zones handle DNS for internal VPC resources (e.g., databases). Use Public Zones for internet-facing services and Private Zones for internal network resolution.

3. Routing Policies:
   - Simple: Directs traffic to a single resource.
   - Weighted: Splits traffic based on assigned weights.
   - Latency: Routes to the region with the lowest latency.
   - Failover: Active-passive setup for disaster recovery.
   - Geolocation: Routes based on user location.
   - Multi-Value: Returns multiple healthy IPs for DNS queries.

4. Health Checks monitor endpoints (e.g., servers, ALBs). If an endpoint fails, Route 53 reroutes traffic to healthy resources, ensuring high availability.

5. Multi-Region Failover Strategy:
   - Use a Failover Routing Policy with primary (active) and secondary (passive) resources (e.g., EC2 in Region A and B).
   - Configure health checks for both regions. If the primary fails, Route 53 automatically redirects traffic to the secondary.

Amazon SQS and SNS
1. Amazon SQS decouples application components by allowing asynchronous communication via message queues. Producers send messages to queues, and consumers retrieve them, enabling independent scaling and fault tolerance.

2. Standard Queues offer high throughput, best-effort ordering, and at-least-once delivery. FIFO Queues guarantee order, exactly-once processing, and limited throughput. Use FIFO for transactional systems (e.g., banking) and Standard for high-volume apps (e.g., logging).

3. Amazon SNS broadcasts messages to multiple subscribers (e.g., email, SMS, Lambda) via topics. Publishers send messages to a topic, and all subscribed endpoints receive them simultaneously.

4. Message Filtering lets subscribers receive only messages matching specific attributes (e.g., "region=us-west"). This reduces overhead by avoiding irrelevant message processing.

5. Ride-Sharing Solution:
   - Use SNS to broadcast real-time trip updates to drivers via mobile push notifications.

- Use SQS FIFO Queues to process ride requests in order, ensuring no duplicates.

Amazon SES
1. Amazon SES is a cost-effective email service for sending transactional (e.g., order confirmations) and marketing emails at scale. It handles reputation management, bounce tracking, and compliance.

2. Transactional Emails are triggered by user actions (e.g., password reset). Marketing Emails are bulk promotions. SES optimizes delivery by separating these streams and managing sender reputation.

3. Sending Quotas limit the number of emails sent per 24-hour period. These are based on account history and reputation. Exceeding quotas may throttle email delivery.

4. SPF validates sender IPs. DKIM adds a digital signature to emails. DMARC defines how to handle emails failing SPF/DKIM checks. Together, they prevent spoofing and improve deliverability.

5. E-commerce Solution:
   - Use SES to send transactional emails (order confirmations) and marketing campaigns.
   - Implement SPF, DKIM, and DMARC for authentication.
   - Use dedicated IPs and monitor bounce rates with SES dashboards.

Virtualization
1. Type 1 Hypervisors (bare-metal) run directly on hardware (e.g., VMware ESXi, AWS Nitro). Type 2 Hypervisors run atop an OS (e.g., VirtualBox).

2. VM Migration involves moving VMs between hosts. Challenges include downtime, network latency, and storage compatibility.

3. Virtualization underpins IaaS by abstracting physical hardware into on-demand virtual resources (e.g., EC2 instances).

4. Virtualization Strategy:
   - Migrate workloads to a Type 1 hypervisor (e.g., VMware) for performance.
   - Use live migration tools (e.g., vMotion) to minimize downtime.
   - Implement auto-scaling to optimize resource usage.

Containerization (Docker)
1. Containerization packages apps and dependencies into isolated, lightweight containers. Unlike VMs, containers share the host OS kernel, reducing overhead.

2. Docker simplifies deployment by standardizing container images, ensuring consistency across environments.

3. Kubernetes automates scaling, load balancing, and self-healing for containerized apps.

4. Use containers for microservices and rapid scaling. Use VMs for legacy apps requiring full OS isolation.

5. Fintech Solution:
- Dockerize microservices and deploy on Kubernetes (e.g., EKS).
- Use Kubernetes for auto-scaling, rolling updates, and monitoring.

CDN (Amazon CloudFront)
1. CloudFront accelerates content delivery via edge locations, caching content closer to users.

2. Origin: Source of content (e.g., S3). Edge Locations: CDN servers. Cache Behavior: Rules for caching (e.g., TTL).

3. Signed URLs/Cookies restrict access to private content (e.g., paid videos).

4. Integration with AWS Shield (DDoS protection) and WAF (traffic filtering) secures content.

5. Media Streaming Solution:
- Create a CloudFront distribution with S3 as the origin.
- Use signed URLs for secure video access.
- Enable AWS WAF and Shield for protection.

Database (Amazon RDS)
1. Amazon RDS simplifies relational database management via automated backups, patching, and scaling.

2. Multi-AZ provides high availability with a standby replica in another AZ. Read Replicas offload read traffic. Use Multi-AZ for critical workloads and Read Replicas for read-heavy apps.

3. Supported Engines: MySQL, PostgreSQL, MariaDB, Oracle, SQL Server. Choose based on licensing, features, and compatibility.

4. Backups/Snapshots enable point-in-time recovery. Automated backups retain data for up to 35 days.

5. E-Commerce Architecture:
- Deploy Multi-AZ RDS for the primary database.
- Add Read Replicas in multiple regions for global read scaling.
- Enable automated backups and use Aurora for higher performance.