AWS Lambda  
  
**Q1. What is AWS Lambda?**

AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers. You upload your code, and Lambda takes care of scaling and managing resources.

**Q2. What are the benefits of using AWS Lambda?**

* Cost-effective: Pay only for resources consumed, eliminating idle server costs.
* Scalability: Automatically scales to handle any traffic volume.
* Faster development: Focus on code without server infrastructure concerns.
* Simplified operations: No need to manage servers or patching operating systems.
* Integration with other AWS services: Seamless integration for various functionalities.

**Q3. What are the limitations of AWS Lambda?**

* Cold start times: Initial delay during the first invocation.
* Limited execution time: Each function has a maximum execution time limit (default 15 minutes).
* Limited memory: Functions have a memory allocation limit determining available resources.

**Q4. What programming languages does AWS Lambda support?**

Lambda supports popular programming languages such as Node.js, Python, Java, C#, Go, Ruby, and PowerShell.

**Q5. How do you trigger AWS Lambda functions?**

Lambda functions can be triggered by events from AWS services like S3, SNS, API Gateway, and more.

**Q6. What is AWS Lambda cold start, and how can it be mitigated?**

Cold start refers to the initial delay during the first function invocation. Mitigation methods include provisioned concurrency, caching, and code optimization.

**Q7. How can you monitor and troubleshoot AWS Lambda functions?**

* CloudWatch: Provides metrics, logs, and tracing information for monitoring and issue identification.
* AWS X-Ray: Offers detailed tracing information for troubleshooting complex deployments.

**Q8. What are some best practices for developing and deploying AWS Lambda functions?**

* Keep functions small and focused.
* Handle errors gracefully with proper error handling.
* Utilize environment variables for sensitive configuration data.
* Test functions thoroughly with unit and integration tests.

**Q9. What are the security considerations for using AWS Lambda?**

* Utilize IAM to control access to Lambda functions and resources.
* Define IAM roles with least privilege principles.
* Avoid storing sensitive data directly in Lambda code.
* Utilize AWS KMS for managing encryption keys.

**Q10. What are some use cases for AWS Lambda?**

* Processing data from S3 buckets (e.g., image resizing, data transformations).
* Responding to API requests through API Gateway.
* Triggering workflows based on events from other AWS services.
* Running background tasks and scheduled jobs.

Dynamo DB

**Q1. What is DynamoDB?**

DynamoDB is a NoSQL database service by AWS, providing fast and scalable performance for storing and retrieving data. It's designed for applications needing high availability, low latency, and flexible data models.

**Q2. What are the benefits of using DynamoDB?**

* Scalability: Easily scales tables based on read and write capacity units.
* Performance: High performance with low-latency data access.
* High Availability: Built-in features for fault tolerance and high availability.
* Flexible Data Model: Supports a document data model with a flexible schema.
* Pay-Per-Use: Pay only for provisioned capacity units used for read and write operations.

**Q3. What are the key components of DynamoDB?**

* Tables: Fundamental storage unit with items and attributes.
* Items: Individual data entries within a table, consisting of key-value pairs.
* Attributes: Pieces of data within an item, with names and values of specific data types.
* Primary Key: Uniquely identifies each item, can be a hash key or a combination of hash and range keys.
* Provisioned Capacity Units: Define read and write throughput capacity for tables.

**Q4. What are the different types of primary keys in DynamoDB?**

* Hash Key: Partitions data across storage segments for quick retrieval based on hash key value.
* Range Key: (Optional) Used with hash key to identify items within a partition for efficient querying.

**Q5. How do you query data in DynamoDB?**

* Point Queries: Fetch specific items using exact primary key values.
* Range Queries: Retrieve multiple items based on hash key value and range key values.
* Secondary Indexes: Optional indexes on non-primary key attributes for efficient querying.

**Q6. How can you manage and monitor DynamoDB tables?**

* AWS Management Console: Web-based interface for table management and monitoring.
* AWS CLI: Programmatic access for automation.
* CloudWatch: Provides metrics on table performance, capacity utilization, and API calls.

**Q7. What are some best practices for designing and using DynamoDB tables?**

* Define a clear data model: Plan table structure, primary keys, and data types based on access patterns.
* Denormalize data: Store frequently accessed data together to minimize read operations.
* Optimize for access patterns: Provision capacity units based on expected throughput.
* Utilize secondary indexes: Create indexes on frequently queried attributes.

**Q8. What are security considerations for using DynamoDB?**

* Utilize IAM: Control access to tables and operations.
* Define least privilege IAM policies.
* Implement encryption at rest and in transit for sensitive data.

**Q9. How does DynamoDB compare to relational databases like RDS?**

* DynamoDB: NoSQL, suitable for high-scale, flexible data models with fast performance.
* RDS: Relational, structured data storage with SQL queries, ideal for applications with complex data relationships.

API Gateway

**Q1. What is API Gateway?**

API Gateway is a fully-managed service by AWS that simplifies building, deploying, managing, and monitoring APIs. It acts as a single entry point for APIs, handling incoming HTTP requests and routing them to backend services.

**Q2. What are the benefits of using API Gateway?**

* Simplified API development: Focus on building API logic without managing server infrastructure or security.
* Scalability: Automatically scales to handle high volumes of traffic.
* Security: Offers built-in features like access control and throttling.
* Performance: Optimizes API calls for low latency and high throughput.
* Monitoring and analytics: Provides insights into API usage and performance.

**Q3. What are the key components of API Gateway?**

* REST APIs: Utilizes HTTP verbs and resources for data access.
* Methods: Define how API resources are accessed (GET, POST, PUT, DELETE).
* Resources: Represent logical entities exposed by the API.
* Integration targets: Backend services handling API requests.
* Stages: Environments where the API is deployed (e.g., development, staging, production).

**Q4. What are the different types of integration targets in API Gateway?**

* AWS Lambda functions: Ideal for serverless API logic.
* Amazon EC2 instances: Run API logic on managed EC2 instances.
* Amazon ECS services: Integrate with containerized applications on ECS.
* HTTP endpoints: Connect with external APIs hosted outside of AWS.

**Q5. How does API Gateway handle authentication and authorization?**

* API Keys: Basic authentication for API access.
* IAM (Identity and Access Management): Granular access control integration.
* Amazon Cognito: Enables user authentication and authorization through a user pool.

**Q6. How can you monitor and analyze traffic through your API Gateway APIs?**

* CloudWatch: Provides metrics and logs for requests, latencies, and errors.
* API Gateway throttling and quotas: Manages API usage and prevents abuse.

**Q7. What are some best practices for designing and deploying APIs with API Gateway?**

* Define clear API resource structures.
* Use appropriate HTTP methods for different actions.
* Implement robust error handling and informative responses.
* Document APIs using OpenAPI Specification (OAS).
* Consider caching mechanisms for frequently accessed resources.

**Q8. How does API Gateway compare to other API development tools?**

* API Gateway: Managed service with built-in features for security, scaling, and monitoring.
* Self-developed APIs: Offers more control but requires managing infrastructure.
* Third-party API frameworks: Provide various functionalities but can add complexity.

**Q9. What are the pricing considerations for using API Gateway?**

* You are charged based on the number of API requests processed.
* Free tier limits are available for exploration and experimentation.

Cognito

**Q1. What is AWS Cognito?**

AWS Cognito is a service that manages user authentication, authorization, and sign-in for web and mobile applications. It provides features such as user registration, login, social sign-in, password management, and token-based authorization.

**Q2. What are the benefits of using Cognito?**

* Simplified user authentication: Offloads user management tasks, allowing you to focus on core functionalities.
* Scalability: Easily scales to handle large numbers of users.
* Security: Provides built-in security features like password hashing, multi-factor authentication (MFA), and user activity monitoring.
* Customization: Offers control over user registration process, sign-in flows, and user attributes.
* Integration with other AWS services: Integrates seamlessly with services like S3, DynamoDB, and IAM.

**Q3. What are the key components of Cognito?**

* User pools: Stores user data like usernames, passwords, and attributes.
* User groups (optional): Organize users into groups for access control.
* Identity pools: Provide temporary AWS credentials to authorized users for accessing AWS resources.
* User sign-in flows: Define how users can sign in to your application (e.g., username/password, social login).

**Q4. What are the different sign-in options with Cognito?**

* Standard username and password authentication.
* Social sign-in with providers like Facebook, Google, and Amazon.
* SAML (Security Assertion Markup Language) integration for enterprise authentication.

**Q5. How does authorization work with Cognito?**

* Cognito issues tokens (JWT - JSON Web Tokens) to authenticated users.
* Applications can validate tokens to determine user identity and access permissions.
* Authorization policies within applications control access to specific resources based on user attributes or groups.

**Q6. Can Cognito integrate with social identity providers (like Facebook or Google)?**

Yes, Cognito allows integration with social identity providers, enabling users to sign in using their existing social media accounts for simplified registration and improved user experience.

**Q7. How can I manage user data in Cognito?**

* Define custom user attributes in your Cognito user pool to store additional user information.
* Cognito provides APIs for managing user data, including adding, updating, and deleting users and their attributes.

**Q8. What are some security considerations for using Cognito?**

* Implement strong password policies and enforce MFA (Multi-Factor Authentication) for additional security.
* Regularly update the Cognito service itself to benefit from the latest security patches.
* Follow secure coding practices in your application to prevent vulnerabilities that could expose user data.

**Q9. What are the costs associated with Cognito?**

Cognito has a free tier that allows for a limited number of monthly active users and monthly sign-in events. Beyond the free tier, costs are based on the number of monthly active users and other usage metrics.