ROUTE 53
SIGNIFICANCE OF ROUTE 53 IN DNS

By –

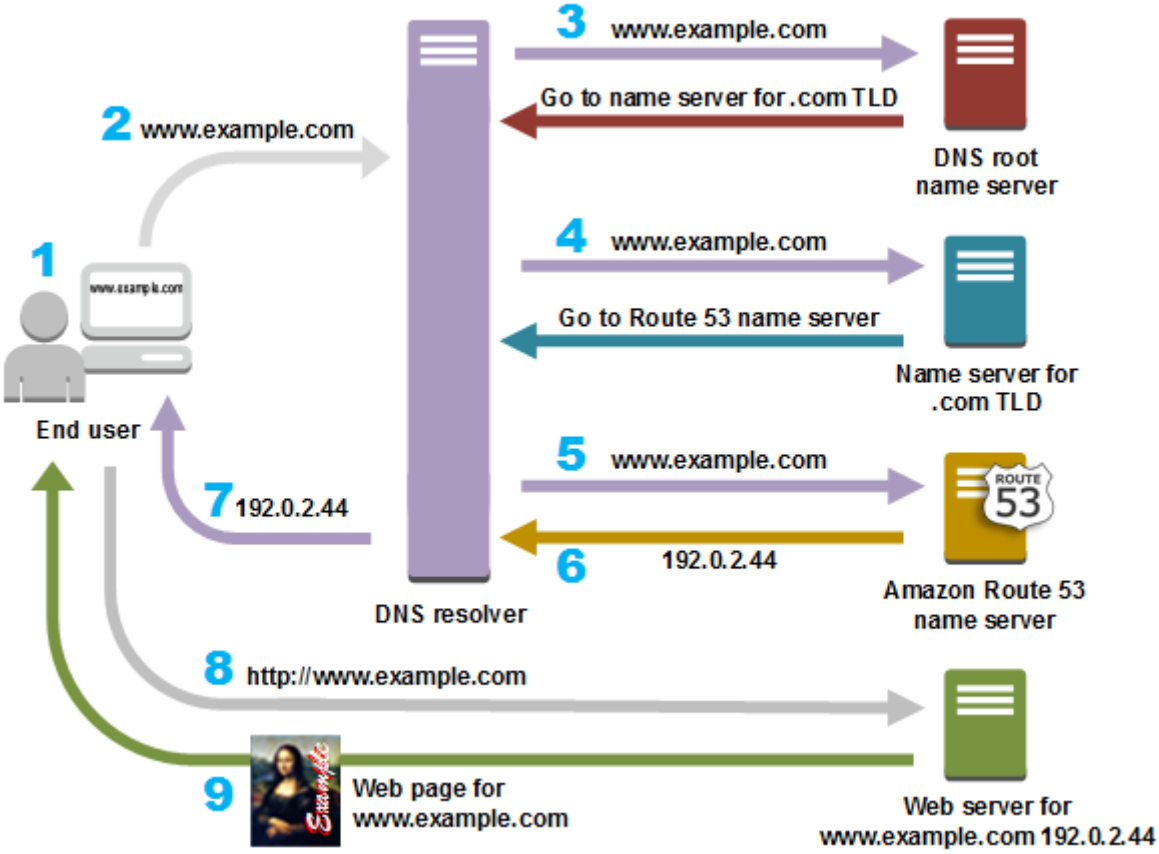
Manichandu .K

Amazon Route 53:

- Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service.
- You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.
- It can be used to do main three functions :
  1) Register domain names
  2) Route internet traffic to the resources for your domain
  3) Check the health of your resources

Purpose of Route 53:

- The purpose of Route 53 service is to register domain names, DNS routing and health checking.
- Route 53 allows you to register domain names for your website or web application.
- When you need a name for your we generally opt for godaddy, cloudflare, clouDNS etc….
- But in AWS the route 53 service can help us get our own DNS and even if you bought DNS outside you can still use it in our AWS console.
- Route 53 will secure that domain.
- When users enter your domain name or subdomain in their web browsers, route 53 ensures that their requests are directed to the appropriate resources.
- It will create a public hosted zone that has the same name as the domain.
- It can efficiently connects the browser of user to your website or application.
- Route 53 periodically checks the health of your resources by sending automated requests over the internet.
- If a resource becomes unavailable, you can receive notifications and choose to route traffic away from unhealthy resources.



Significance of DNS in context of ROUTE-53:

- DNS resolver – DNS translates human-friendly domains (example.com) into IP addresses (like 192.0.2.44).
- Route 53 acts as a DNS service, ensuring efficient and accurate name resolution. It maps domain names to the appropriate ip addresses of servers or resources.
- Route 53 routes incoming requests to the right resources based on DNS records.
- You can configure route 53 to direct traffic to different endpoints based on geographic location using the DNS records.
- Route 53 integrates seamlessly with other AWS services like Elastic Load Balancers (EBS) and S3.
- Route 53 supports DNSSEC(domain name system security extensions) which adds extra layer of security to DNS.

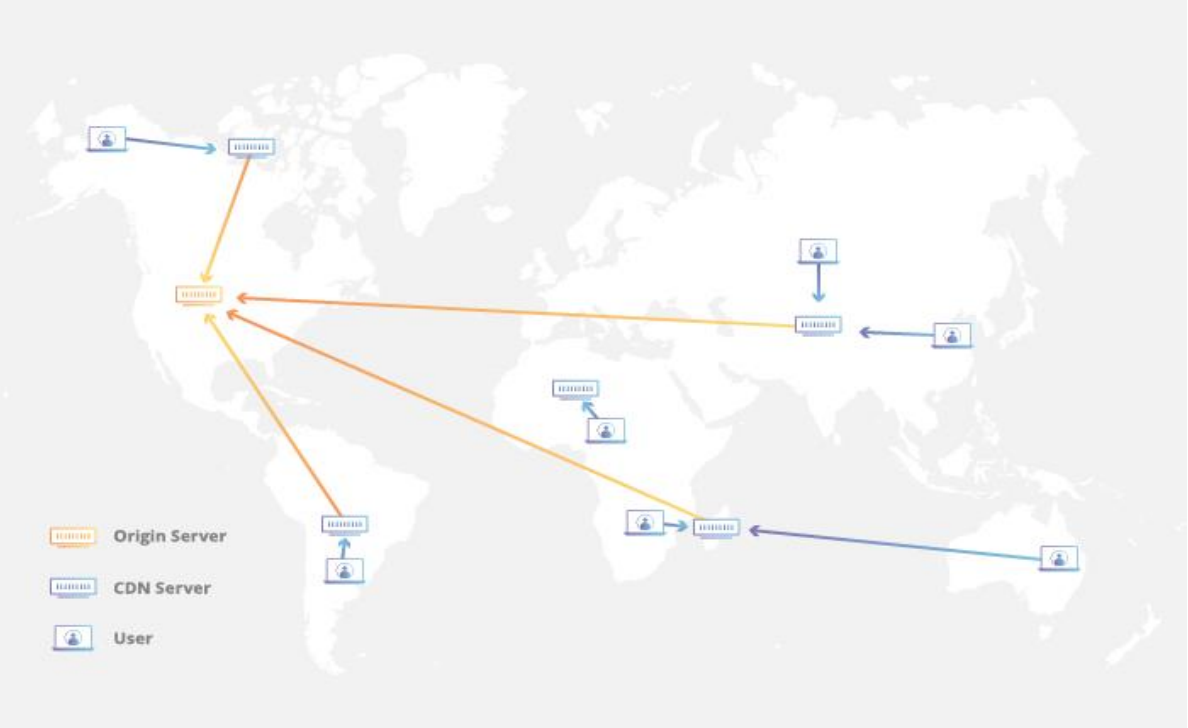
AMAZON CLOUD FRONT

By-

Manichandu .K

Amazon CloudFront:

- CloudFront is an integral part of Amazon web services which is a robust Content Delivery Network (CDN) designed to optimize and expedite the distribution of web videos, bulky media and other file types to your audience.
- It gives businesses and web application developers an easy and cost effective way to distribute content with low latency and high data transfer speeds.
- Like any other service in AWS, cloudfront is self-service, pay-per-use offering, requiring no long term commitments for minimum fees.
- With cloudfront, your files are delivered to end-users using a global network of edge locations.
- The key features of cloudfront are :
  1) Content delivery network
  2) Edge locations
  3) Low latency and high transfer speeds
  4) Edge compute
  5) Security and encryption
  6) Original shield
- Content delivery network (CDN): when users request content, cloudfront delivers it from nearest edge location, minimizing round-trip time.
- Edge locations: cloudfront operates through a worldwide network of data centers called edge locations. These edge locations are strategically distributed across the globe, ensuring efficient delivery of content to end-users.
- CloudFront caches the data from your main server to these datacentres called edge locations so that they can be accessed with very low latency.
- They also support secure connections via HTTPS (SSL/TLS).it also provides protection against DDoS attacks (distributed denial of service).



Critical challenges related to content delivery, scalability and performance:

- Latency and speed – traditional web hosting relies on a single server location, leading to latency for users far from that server.
- Scalability and high traffic – handling sudden spikes in traffic can overwhelm a single server.
- Global reach – serving content to a global audience requires infrastructure in various regions.
- Content caching – frequent requests for the same content strain origin servers.
- Origin shield – frequent cache misses put strain on the origin server.

CloudFront's edge location system and caching effectively address all the challenges mentioned above.

Cloudfront can also integrate with Lambda@Edge allowing custom code execution at edge locations for dynamic content optimization.

SERVERLESS COMPUTING – LAMBDA

By-

Manichandu .K

Lambda Function:

Serverless computing has revolutionized the way we build and deploy applications. Among the leading serverless platforms, AWS Lambda stands out as a powerful compute service provided by Amazon Web Services (AWS). In this document, we delve into the intricacies of AWS Lambda, exploring its features, use cases, and best practices. Whether you're a developer, architect, or business owner, understanding AWS Lambda is essential for building efficient, scalable, and cost-effective applications.

1. Introduction to Serverless Computing

Serverless computing shifts the focus from managing servers to writing code. AWS Lambda exemplifies this paradigm, allowing developers to execute code in response to events without worrying about infrastructure.

2. AWS Lambda: The Compute Layer

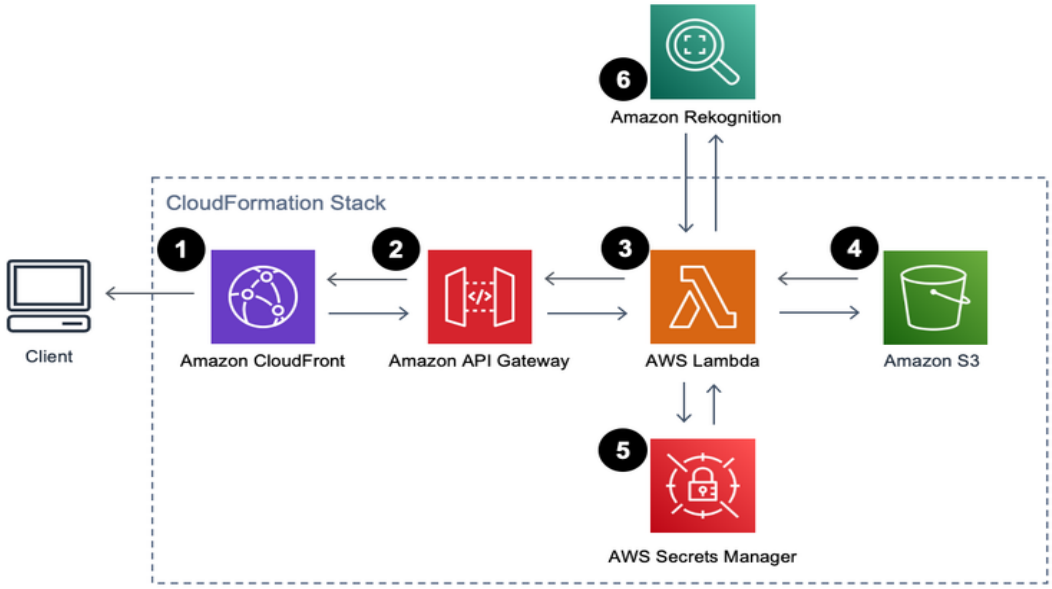
AWS Lambda operates on an event-driven model. Developers create functions, upload code, and let Lambda handle scaling, monitoring, and maintenance. Billing is based on actual compute time used.

3. Use Cases for AWS Lambda

Lambda serves various purposes:

- Microservices: Break down applications into smaller functions.
- Data Processing: Handle resource-intensive tasks.
- Serverless APIs: Build APIs without server management.
- Real-Time File Processing: Process uploads, generate thumbnails.

- IoT Applications: Respond to device events.
- Custom Logic: Execute business-specific code.



4. Environment Variables in AWS Lambda

Environment variables allow dynamic configuration:

- Adjust behaviour without code changes.
- Securely store sensitive information.
- Customize based on environments (dev, prod).

5. Best Practices for AWS Lambda

- Avoid Hardcoding: Use environment variables.
- Secrets Management: Leverage AWS Secrets Manager.
- Optimize Memory Size: Right-size your functions.
- Provisioned Concurrency: Handle high demand efficiently.

6. Case Studies and Customer Stories

- Capital One: Reduced costs and saved developer time.
- DISCO: Improved search results using Lambda.
- Honeycomb: Enhanced performance and efficiency.

In conclusion, AWS Lambda empowers developers to build efficient, scalable applications without managing servers. Its integration with environment variables further enhances flexibility and security.