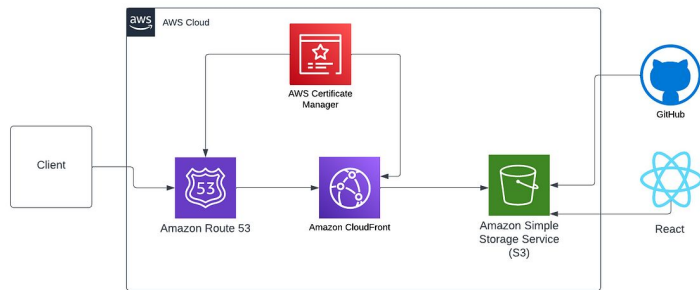


Simple Storage Service (S3)

Amazon Simple Storage Service (S3) is an object storage service designed for scalability, durability (99.999999999%), and availability. It allows you to store and retrieve any amount of data from anywhere on the web.

- Storage Classes: Includes Standard, Intelligent-Tiering, Glacier, and Glacier Deep Archive to optimize cost for different access patterns.
- Durability and Availability: Data is automatically replicated across multiple Availability Zones.
- Security: Supports bucket policies, IAM policies, encryption at rest (SSE-S3, SSE-KMS) and in transit (HTTPS).
- Versioning & Lifecycle: Allows version control of objects and lifecycle rules to transition or delete objects.
- Event Notifications: Can trigger AWS services like Lambda when specific events occur (e.g., object upload).
- Static Website Hosting: You can host websites directly from an S3 bucket.
- Access Control: Uses IAM, ACLs, bucket policies, and pre-signed URLs to control access.



S3 Bucket Policy

An **S3 Bucket Policy** is a JSON-based resource policy attached directly to an S3 bucket. It controls access by defining who can perform which actions on the bucket and its contents. Bucket policies support fine-grained permissions, cross-account access, and condition-based rules.

- **Written in JSON** format and attached to the bucket (not IAM user or role).
- **Defines Effect** (Allow/Deny), Principal, Action, Resource, and optional Condition.
- Use Cases:
 - Granting public read access (e.g., for static websites).
 - Allowing another AWS account access to the bucket.
 - Denying access unless using HTTPS (via `aws:SecureTransport`).

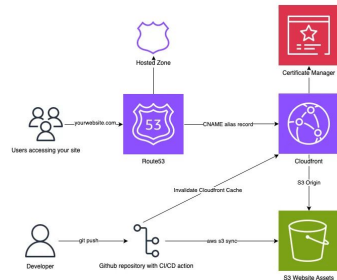
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3::your-bucket-name/*"
    }
  ]
}
```

S3 Static Website Hosting

Amazon S3 allows you to host static websites (HTML, CSS, JS only – no server-side logic) directly from a bucket.

You must enable static website hosting in the bucket properties.

- You define:
 - Index document (e.g., index.html)
 - Optional error document (e.g., error.html)
- The bucket must have a bucket policy allowing public s3:GetObject access.
- The "Block all public access" setting must be disabled.
- The website is served via an HTTP endpoint (not HTTPS unless served through CloudFront).
- Use Route 53 + CloudFront for a custom domain and HTTPS.



S3 Versioning

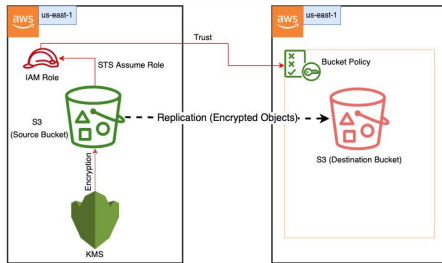
Amazon S3 Versioning enables you to preserve, retrieve, and restore every version of every object stored in a bucket.

- Enablement: Must be explicitly enabled per bucket. Once enabled, it cannot be disabled—only suspended.
- Versioning: Each new upload of the same object key creates a new version with a unique version ID.
- Accidental Deletion Protection: A delete action adds a delete marker instead of removing the object. Older versions remain accessible.
- MFA Delete: Adds an extra layer of protection by requiring multi-factor authentication for version deletions.
- Cost Management: Works with lifecycle policies to transition or delete old versions and control storage costs.

S3 Replication

Amazon S3 Replication allows you to automatically copy objects across buckets—either within the same AWS Region or to another Region—for backup, compliance, or low-latency access.






- Cross-Region Replication (CRR): Replicates to a bucket in another region.
- Same-Region Replication (SRR): Replicates to another bucket in the same region.
- Versioning must be enabled on both source and destination buckets.
- The IAM role used must have permission to replicate objects.
- Replication does not apply retroactively to existing objects unless explicitly configured.
- Can replicate tags, ACLs, metadata, and encrypted objects (if using SSE-KMS with appropriate permissions).
- Option to replicate only specific prefixes or object tags.
- Can be configured to delete replicated objects when the source is deleted (delete marker replication).



S3 Storage Classes

Amazon S3 offers multiple storage classes optimized for different use cases, balancing cost, availability, and access frequency.

- **S3 Standard**: For general-purpose storage of frequently accessed data. High durability, low latency, and high throughput.
- **S3 Intelligent-Tiering**: Automatically moves data between access tiers based on usage patterns. Ideal when access frequency is unpredictable.
- **S3 Standard-IA (Infrequent Access)**: For data that is less frequently accessed but requires rapid access when needed. Lower cost than Standard.
- **S3 One Zone-IA**: Similar to Standard-IA but stored in a single AZ. Lower cost, but lower availability and no AZ redundancy.
- **S3 Glacier**: Low-cost archival storage. Retrieval times range from minutes to hours. Suitable for long-term backups.
- **S3 Glacier Deep Archive**: Lowest-cost storage class, designed for long-term data archiving with retrieval times of up to 12 hours.

					
S3 Standard	S3 Intelligent-Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Frequent	Access frequency			Archive	
<ul style="list-style-type: none"> • Active, frequently accessed data • Milliseconds access • ≥ 3 AZ • \$0.0210/GB 	<ul style="list-style-type: none"> • Data with changing access patterns • Milliseconds access • ≥ 3 AZ • \$0.0210 to \$0.0125/GB • Monitoring fee per object • Min storage duration 	<ul style="list-style-type: none"> • Infrequently accessed data • Milliseconds access • ≥ 3 AZ • \$0.0125/GB • Retrieval fee per GB • Min storage duration • Min object size 	<ul style="list-style-type: none"> • Re-creatable, less accessed data • Milliseconds access • 1 AZ • \$0.0100/GB • Retrieval fee per GB • Min storage duration • Min object size 	<ul style="list-style-type: none"> • Archive data • Select minutes or hours • ≥ 3 AZ • \$0.0040/GB • Retrieval fee per GB • Min storage duration 	<ul style="list-style-type: none"> • Long-term archive-data • Select hours • ≥ 3 AZ • \$0.00099/GB • Retrieval fee per GB • Min storage duration

S3 Exam Questions

A company wants to replicate critical data from a bucket in us-east-1 to eu-west-1 for compliance reasons. They also want to reduce storage costs by archiving data that is rarely accessed.

Which configuration meets the requirements with minimal operational overhead?

- A. Enable S3 Same-Region Replication and use S3 One Zone-IA.
- B. Enable S3 Cross-Region Replication and set a lifecycle rule to transition objects to Glacier.
- C. Enable S3 Cross-Region Replication and manually copy objects using the AWS CLI.
- D. Use S3 Standard and download the data periodically to archive it on-premises.

An organization enables Cross-Region Replication (CRR) for compliance. After configuring it, they notice that objects uploaded before replication was set up are not copied to the destination bucket.

What is the most likely reason?

- A. Versioning is not enabled on the destination bucket.
- B. IAM permissions are incorrectly configured.
- C. Replication does not apply to existing objects by default.
- D. The source bucket is in the same region as the destination bucket.

Route 53

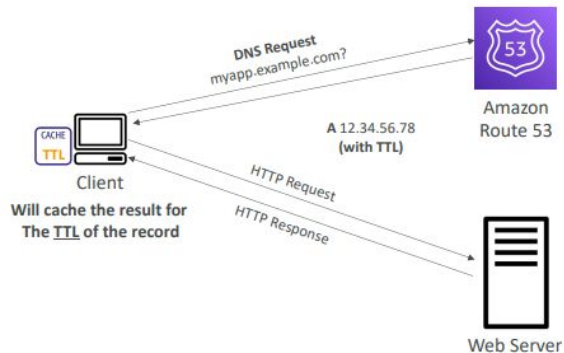
Amazon Route 53 is a highly available and scalable Domain Name System (DNS) service. It is used to register domain names, route internet traffic to AWS or external resources, and check the health of resources.

- Domain Registration – Allows purchasing and managing domain names directly in Route 53.
- DNS Service – Translates human-readable domain names (e.g., example.com) into IP addresses.
- Health Checks – Monitors the health of endpoints and can redirect traffic if a target is unhealthy.
- Routing Policies – Supports multiple policies for routing decisions:
 - Simple
 - Weighted
 - Latency-based
 - Failover
 - Geolocation
 - Multivalue answer
- Global Availability – Built on a worldwide network of DNS servers for high reliability and low-latency responses.
- AWS Integration – Seamlessly integrates with other AWS services such as ELB, CloudFront, and S3.

Route 53 – TTL

TTL (Time to Live) in Amazon Route 53 defines the amount of time (in seconds) that DNS resolvers cache the record before querying Route 53 again.

- **Purpose** – Helps control how frequently DNS changes propagate to end users.
- **Lower TTL** – Useful when expecting frequent DNS changes (e.g., failover scenarios), but increases the number of DNS queries.
- **Higher TTL** – Reduces DNS query traffic and improves performance, but delays propagation of updates.
- **Default Setting** – TTL is user-defined per record; common values range from 60 seconds to several hours depending on use case.
- **Impact on Failover** – For routing policies like failover or weighted routing, setting an appropriate TTL is crucial to ensure timely redirection.



CNAME vs Alias

- CNAME (Canonical Name Record):
Points a domain name to another domain name
(e.g., app.example.com → app.mycompany.com).

Used for: Redirecting one domain to another.

Restriction: Cannot be used at the root domain level
(e.g., you can't use CNAME for example.com, only for subdomains).

Standard DNS record: Works with any DNS provider.

When to use which?

Use CNAME when pointing to a non-AWS domain and on subdomains.

Use Alias when integrating with AWS services or pointing a root domain to AWS.

- Alias Record:
A Route 53-specific feature that allows pointing a domain name (even root domain) to **AWS resources like:**

S3 buckets configured as static websites, CloudFront distributions, ELB load balancers, API Gateway custom domains.

Used for: AWS resource integration.

Can be used at the root domain level (e.g., example.com) – unlike CNAME.

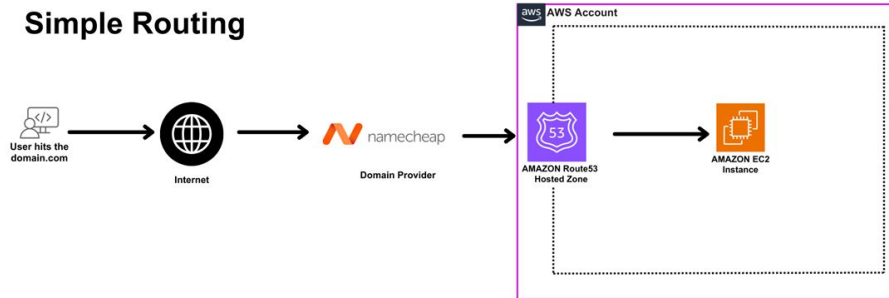
No extra DNS lookup: Alias records resolve to the AWS IP directly, improving performance.

Simple Routing Policy

The Simple routing policy in Amazon Route 53 is the most basic type of routing. It is used when you want to associate a single resource (like a web server or an S3 static website) with a domain name.

- **Single record per name** – You can only have one record for each DNS name with this policy.
- **No special routing logic** – Route 53 simply responds with the IP or resource assigned to the domain.
- **No health checks** – Unless used with a single record and health check explicitly configured.
- **Use case** – Ideal when you have one web server or endpoint handling all traffic for a domain (e.g., `www.example.com` → EC2 instance).

Simple routing is commonly used for straightforward applications without failover, load balancing, or location-based requirements.



Weighted Routing Policy

The Weighted routing policy in Amazon Route 53 lets you distribute traffic across multiple resources based on assigned weights.

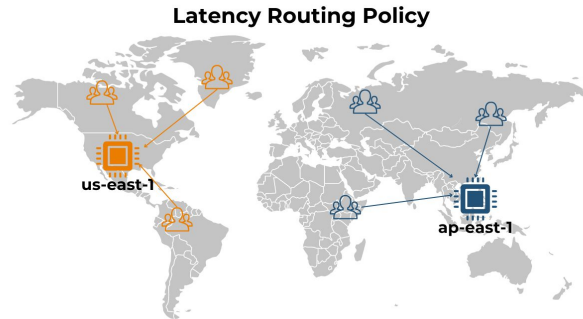
- **Multiple records** for the same domain name – Each with a different weight.
- **Weights determine traffic** distribution – For example, 70:30 will route ~70% of traffic to one resource and 30% to the other.
- **Weights are relative** – They don't need to add up to 100 (e.g., 1 and 9 still give a 10%/90% split).
- **Optional health checks** – Can associate health checks to route traffic only to healthy endpoints.
- **Use case** – Useful for load testing, gradual traffic shifting during deployments, or A/B testing.



Latency-Based Routing Policy

The Latency-based routing policy in Amazon Route 53 directs user traffic to the AWS Region that provides the lowest latency (fastest response time) for the user's location.

- **Works across multiple AWS Regions** – You can associate records with resources in different regions.
- **Improves performance** – Users are routed to the region that minimizes network delay.
- **Requires multiple resources** – Each record must specify a different AWS region.
- **Optional health checks** – Can be used to ensure only healthy resources receive traffic.
- **Use case** – Ideal for global applications hosted in multiple AWS regions, where user experience depends on low latency.



Route 53 Health Checks

Amazon Route 53 health checks monitor the availability and performance of resources (e.g., web servers, endpoints) and help route traffic only to healthy targets.

- Types of health checks – Can monitor:

Public endpoints (IP address or domain name)

CloudWatch alarms

Other health checks (calculated health checks)

- Failure threshold – You can set how many consecutive failures count as unhealthy.
- Interval – Can be configured (typically 30s or 10s for fast checks).
- Used with routing policies – Especially effective with: Failover, Weighted, Multivalue answer.
- Notifications – Can integrate with CloudWatch and SNS to alert when a target becomes unhealthy.
- Failover support – Automatically redirects traffic to a healthy resource if the primary fails.

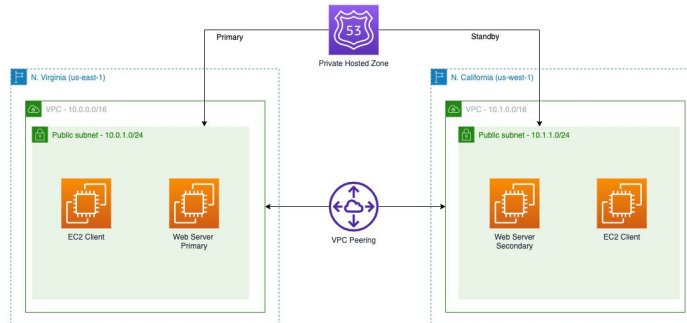
Health checks are key to building highly available architectures with smart routing decisions.

Failover Routing Policy

The Failover routing policy in Amazon Route 53 is used to automatically redirect traffic from a primary resource to a secondary (backup) resource in case of failure.

- **Two record sets only** – One set as Primary, the other as Secondary.
- **Health check required** on the primary – Route 53 monitors its status.
- **Automatic failover** – If the primary becomes unhealthy, Route 53 routes traffic to the secondary.
- **Restores traffic to primary** – Once the primary becomes healthy again.
- **Use case** – Ideal for high availability setups, such as backing up an EC2 website with an S3 static site.

Failover routing ensures service continuity by rerouting users to an available backup automatically

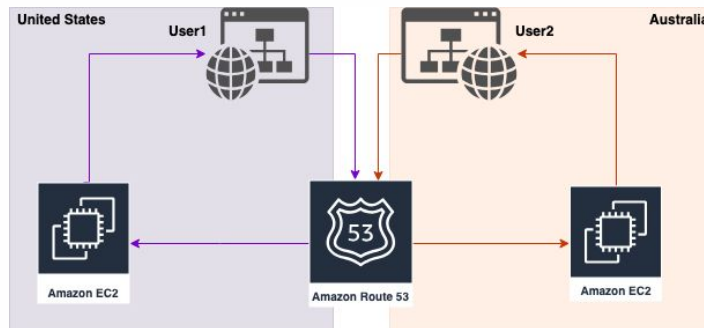


Geolocation Routing Policy

The Geolocation routing policy in Amazon Route 53 routes traffic based on the geographic location of the user making the DNS request.

- **Routes by location** – You can define responses for users based on: Continent, Country.
- **Each location gets a specific record** – You assign different endpoints for each geographic region.
- **Default record** – You can configure a default route for users that don't match any specified location.
- **Use case** – Common for regulatory compliance, regional language content, or directing traffic to the nearest regional office or server.

Geolocation routing gives control over where user traffic is sent, based on where the user is located, not where the resources are hosted.

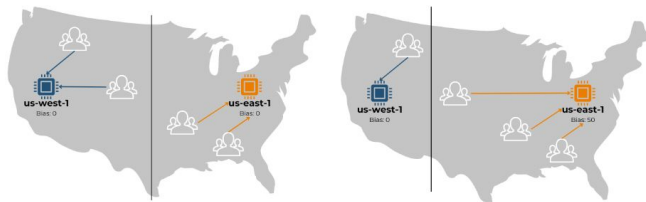


Geoproximity Routing Policy

The Geoproximity routing policy in Amazon Route 53 (used with Route 53 Traffic Flow) routes traffic based on the location of users and the geographic location of your AWS resources — with optional biasing to shift traffic.

- **Requires Route 53 Traffic Flow** – Managed via visual policies (not regular hosted zone records).
- **Routes by distance** – Traffic is routed to the AWS resource (e.g., EC2, ELB) closest to the user's location.
- **Supports bias values** – You can shift more traffic to a specific location by increasing or decreasing its "radius of influence."
- **Supports both AWS and non-AWS locations** – You can use latitude/longitude coordinates.
- **Use case** – Useful for performance optimization while maintaining geographic control.

Geoproximity routing gives fine-tuned control over regional traffic distribution, combining location-based decisions with custom influence settings.



IP-based Routing

Routing is based on the client's IP address.

You provide a list of CIDR blocks that represent your clients, and map each to a specific endpoint or location (user-IP-to-endpoint mapping).

Use cases include:

Optimizing performance

Reducing network costs

Example: Route users from a particular ISP to a specific EC2 instance or service endpoint.

Multi-Value Routing

Purpose: Similar to a simple routing policy, but allows returning multiple values (up to 8 IP addresses) in a single DNS response.

Use Case: Commonly used for basic load balancing and improved availability without using a load balancer.

Health Checks: Route 53 can associate health checks with each IP address. Only healthy IPs will be returned in the DNS response.

Client Behavior: DNS resolvers may randomly select one of the returned IP addresses, distributing traffic.

Example:

You configure a domain example.com with 4 different IP addresses. If 1 of them fails a health check, Route 53 returns only the 3 healthy IPs to the client.

Using a Third-Party Domain with Route 53

Use Case: When your domain is registered with a provider other than Route 53 (e.g., GoDaddy, Namecheap), but you want to manage DNS using Route 53.

Key Concepts: Create a Hosted Zone: In Route 53, create a public hosted zone for your domain (e.g., example.com).

Update Name Servers:

- Route 53 assigns 4 name servers (NS records).
- Go to your third-party domain registrar and replace the existing name servers with the 4 NS values provided by Route 53.

DNS Management: Once updated, all DNS queries will be handled by Route 53.

Why Use This?

Gain access to Route 53's routing policies, health checks, latency-based routing, and integration with AWS services — even if your domain is not registered with AWS.

Route 53 Exam Questions

Your website hosted in AWS must remain highly available. You have two EC2 instances in different regions. You want to route traffic to the primary instance and only route to the secondary instance if the primary fails a health check. Which Route 53 routing policy should you choose?

- A. Simple routing
- B. Geoproximity routing
- C. Failover routing
- D. Multi-value answer routing

A startup has a static website hosted on Amazon S3. They want to make the site publicly accessible at www.example.com. The domain is registered with a third-party registrar. What steps should they take to use Route 53 for DNS management?

- A. Migrate the domain to Route 53 and use a CNAME to S3
- B. Create an Alias record in Route 53 and update name servers at the registrar
- C. Use a latency-based policy with S3 and Route 53
- D. Assign an Elastic IP to the S3 bucket and update the A record