**AWS Route 53 Routing Policies - Complete Guide**

**Table of Contents**

1. Introduction to Route 53
2. Routing Policy Types
3. Failover Policy
4. Latency Policy
5. Weighted Policy
6. Geolocation Policy
7. Health Checks
8. Step-by-Step Implementation
9. Real-World Use Cases
10. Cleanup

**1. Introduction to Route 53**

* **What is Route 53?**  
  AWS's DNS service that translates domain names (e.g., www.example.com) to IP addresses.
* **Key Features**:
  + High availability (99.99% SLA)
  + Multiple routing policies
  + Integrated health checks

**2. Routing Policy Types**

| **Policy** | **Use Case** | **How It Works** |
| --- | --- | --- |
| **Simple** | Basic routing (default) | Returns IP in random order |
| **Failover** | Disaster recovery | Routes to secondary if primary fails |
| **Latency** | Performance optimization | Routes to lowest-latency region |
| **Weighted** | A/B testing, canary deployments | Distributes traffic based on weights |
| **Geolocation** | Regional restrictions | Routes based on user location |

**3. Failover Policy**

**How It Works**

1. **Primary Server**: Handles all traffic when healthy.
2. **Secondary Server**: Takes over if primary fails (health check fails).
3. **Health Checks**: Route 53 monitors server status every 10-30 seconds.

**Implementation Steps**

1. **Create Health Checks**:
   * Go to **Route 53 → Health Checks → Create**.
   * Configure TCP/HTTP checks on port 80.
2. **Create Failover Records**:
   * Primary record (e.g., US server) → Mark as "Primary".
   * Secondary record (e.g., Mumbai server) → Mark as "Secondary".
3. **Test**:
   * Stop primary server → Traffic automatically shifts to secondary.

bash

*# Example: Stop NGINX on primary server*

systemctl stop nginx

**4. Latency Policy**

**How It Works**

* Routes users to the AWS region with the lowest latency.
* **Example**: A user in Europe gets routed to eu-west-1 instead of us-east-1.

**Implementation Steps**

1. **Create Latency Records**:
   * For each region (e.g., us-east-1, ap-south-1), create a record.
   * Route 53 selects the fastest region based on user location.
2. **Verify**:
   * Users in India → Mumbai server.
   * Users in US → North Virginia server.

**5. Weighted Policy**

**How It Works**

* Distributes traffic based on assigned weights (0-255).
* **Use Cases**:
  + **A/B Testing**: 10% traffic to new version, 90% to old.
  + **Load Balancing**: 50% to US, 50% to India.

**Implementation Steps**

1. **Create Weighted Records**:
   * US server: Weight = 1 (minimal traffic).
   * Mumbai server: Weight = 254 (most traffic).
2. **Verify**:
   * ~99% traffic goes to Mumbai, ~1% to US.

**6. Geolocation Policy**

**How It Works**

* Routes traffic based on user's geographic location.
* **Example**:
  + Users in India → Mumbai server.
  + Users in US → North Virginia server.
  + Users in Europe → Frankfurt server.

**Implementation Steps**

1. **Create Geolocation Records**:
   * Assign IPs to specific continents/countries.
2. **Test**:
   * Use VPN to simulate traffic from different regions.

**7. Health Checks**

* **Purpose**: Monitor server health (HTTP/TCP).
* **Settings**:
  + **Interval**: 10-30 seconds.
  + **Threshold**: Number of failures before marking unhealthy.
* **Key Use**: Triggers failover policies.

**8. Step-by-Step Implementation**

**Prerequisites**

* 2 EC2 instances (e.g., US + Mumbai) with NGINX.
* Domain in Route 53 (e.g., cloudvishwakarma.in).

**Steps**

1. **Create Health Checks**:
   * For US server (54.123.141.50:80).
   * For Mumbai server (52.223.231.74:80).
2. **Configure Policies**:
   * **Failover**: Primary (US), Secondary (Mumbai).
   * **Latency**: US + Mumbai records.
   * **Weighted**: US (Weight=1), Mumbai (Weight=254).
3. **Test**:
   * curl failover.cloudvishwakarma.in → Should route to primary.
   * curl latency.cloudvishwakarma.in → Routes to lowest-latency region.
   * curl weighted.cloudvishwakarma.in → Mostly routes to Mumbai.

**9. Real-World Use Cases**

| **Scenario** | **Policy** | **Benefit** |
| --- | --- | --- |
| **Disaster Recovery** | Failover | Zero downtime during outages |
| **Global Apps** | Latency | Faster response times |
| **Canary Deployments** | Weighted | Gradual feature rollouts |
| **Compliance** | Geolocation | Restrict access by country |

**10. Cleanup**

1. **Delete Route 53 Records**.
2. **Terminate EC2 Instances**.
3. **Delete Health Checks**.