**Send SMS using AWS SNS Without a Topic in Spring Boot**

If you want to send **SMS directly to a phone number without using an SNS topic**, follow these steps:

**1. Add Dependencies in pom.xml**

Add the **AWS SDK for SNS** and **Spring Boot dependencies**:

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- AWS SDK for SNS -->

<dependency>

<groupId>software.amazon.awssdk</groupId>

<artifactId>sns</artifactId>

<version>2.20.0</version>

</dependency>

<!-- SLF4J Logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

</dependency>

</dependencies>

**2. Configure AWS Credentials**

Ensure you have valid **AWS credentials** set up.

**Option 1: Use application.properties**

aws.accessKey=YOUR\_AWS\_ACCESS\_KEY

aws.secretKey=YOUR\_AWS\_SECRET\_KEY

aws.region=us-east-1

**Option 2: Use IAM Role (For AWS Deployment)**

If your app runs on **EC2, ECS, or Lambda**, AWS SDK will **automatically use the IAM role** assigned to the instance.

**3. Create the SNS Service Class**

Create a **service class** to send SMS directly to a phone number.

package com.example.snsdemo.service;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.sns.SnsClient;

import software.amazon.awssdk.services.sns.model.PublishRequest;

import software.amazon.awssdk.services.sns.model.PublishResponse;

@Service

public class SnsSmsService {

private final SnsClient snsClient;

public SnsSmsService(

@Value("${aws.accessKey}") String accessKey,

@Value("${aws.secretKey}") String secretKey,

@Value("${aws.region}") String region

) {

this.snsClient = SnsClient.builder()

.region(Region.of(region))

.credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKey, secretKey)))

.build();

}

public String sendSms(String phoneNumber, String message) {

try {

PublishRequest request = PublishRequest.builder()

.message(message)

.phoneNumber(phoneNumber) // Direct SMS without Topic

.build();

PublishResponse response = snsClient.publish(request);

return "SMS sent successfully! Message ID: " + response.messageId();

} catch (Exception e) {

return "Error sending SMS: " + e.getMessage();

}

}

}

**4. Create a REST Controller**

Expose an API endpoint to send SMS.

package com.example.snsdemo.controller;

import com.example.snsdemo.service.SnsSmsService;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/sns")

public class SnsController {

private final SnsSmsService snsSmsService;

public SnsController(SnsSmsService snsSmsService) {

this.snsSmsService = snsSmsService;

}

@PostMapping("/sendSms")

public String sendSms(@RequestParam String phoneNumber, @RequestParam String message) {

return snsSmsService.sendSms(phoneNumber, message);

}

}

**5. Test the API**

Run the **Spring Boot** application and test the API using **Postman or cURL**.

**Postman Request:**

* **Method:** POST
* **URL:** http://localhost:8080/sns/sendSms
* **Body (x-www-form-urlencoded):**
  + phoneNumber = +11234567890
  + message = Hello, this is a test SMS from AWS SNS!

**cURL Command:**

curl -X POST "http://localhost:8080/sns/sendSms?phoneNumber=%2B11234567890&message=Hello%20AWS%20SNS!"

**6. Important Notes**

✅ **Phone Number Format:** Use **E.164 format** (e.g., +11234567890).  
✅ **AWS SNS SMS Limits:** Some AWS regions may have restrictions on sending SMS to certain countries.  
✅ **Pricing:** AWS **charges per SMS** sent. Check AWS SNS pricing in your region.  
✅ **IAM Policy:** Ensure your IAM user has **AmazonSNSFullAccess** permission.

**7. Next Steps**

* ✅ Deploy on **AWS EC2, ECS, or Lambda**.
* ✅ Implement **Rate Limiting** to prevent excessive SMS costs.
* ✅ Add **Logging and Error Handling** for better monitoring.

Would you like to integrate this with **DynamoDB or SQS for message tracking? 🚀**