Here’s how you can **send SMS using AWS SNS Topic in a Spring Boot application** step by step.

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

Ensure you have the required AWS SDK dependencies in your **pom.xml**:

<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**

Make sure your AWS credentials are set up in **application.properties** or **use IAM roles** if deploying on AWS.

**Option 1: Using application.properties**

aws.accessKey=YOUR\_AWS\_ACCESS\_KEY

aws.secretKey=YOUR\_AWS\_SECRET\_KEY

aws.region=us-east-1

aws.sns.topic.arn=arn:aws:sns:us-east-1:123456789012:MySNSTopic

**Option 2: Using IAM Role (if running on AWS)**

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

**3. Create the SNS Service Class**

Create a service class SnsSmsService.java to send SMS using an SNS topic.

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;

private final String snsTopicArn;

public SnsSmsService(

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

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

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

@Value("${aws.sns.topic.arn}") String snsTopicArn

) {

this.snsClient = SnsClient.builder()

.region(Region.of(region))

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

.build();

this.snsTopicArn = snsTopicArn;

}

public String sendSms(String message) {

try {

PublishRequest request = PublishRequest.builder()

.message(message)

.topicArn(snsTopicArn) // Sending SMS via SNS 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**

Create SnsController.java to expose an 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 message) {

return snsSmsService.sendSms(message);

}

}

**5. Test the API**

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

**Postman Request:**

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

**cURL Command:**

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

**6. Confirm Subscription for SMS**

* **AWS SNS Topics require manual confirmation for SMS subscribers**.
* **Go to AWS SNS Console → Subscriptions**.
* If the phone number is **Pending Confirmation**, **manually confirm** it.

**7. When to Use SNS Topics for SMS?**

✅ **Use SNS Topics** if sending SMS to **multiple subscribers** at once (e.g., alerts, notifications).  
✅ **Use Direct SMS** if sending messages to **individual numbers dynamically**.

**Next Steps**

* ✅ Deploy on AWS **EC2 or ECS** using an **IAM Role**.
* ✅ Enable **SNS FIFO Topics** for ordered messages.
* ✅ Add **Exception Handling** for better error management.

Would you like to integrate this with AWS Lambda or SQS as well? 🚀

To **send SMS using AWS SNS Topic in a Spring Boot application**, there are a few **prerequisites** and setup steps you need to follow before jumping into the code.

**✅ Prerequisites**

**1. AWS Account**

You must have an **active AWS account**. If you don’t have one, create it at [aws.amazon.com](https://aws.amazon.com/).

**2. Create an IAM User with SNS Permissions**

* Go to the **IAM** section in the AWS Console.
* Create a new **IAM user** or use an existing one.
* Attach the **AmazonSNSFullAccess** policy (or a custom policy with sns:Publish permission).
* **Generate access key and secret key**. You'll use these in your Spring Boot app.

**3. Create an SNS Topic**

* Go to **Simple Notification Service (SNS)** in the AWS console.
* Click **Create topic**.
* Choose **Standard** or **FIFO** topic (usually Standard for SMS).
* Name your topic, and create it.
* **Copy the Topic ARN**, you’ll need this in your application.

**4. Subscribe a Phone Number to the Topic**

* In the SNS Topic page, choose **Create subscription**.
* Set **Protocol** to SMS.
* Enter a valid phone number (with country code, e.g., +919876543210).
* The user may need to **opt-in** (they will get a confirmation SMS).

**5. Spring Boot Setup**

You’ll need a Spring Boot project with the following dependencies:

* Spring Boot Starter
* AWS Java SDK for SNS

Add this dependency in your pom.xml if you're using Maven:

<dependency>

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

<artifactId>sns</artifactId>

</dependency>

**✅ Summary**

Before sending SMS via AWS SNS from Spring Boot, make sure you have:

* AWS account and credentials
* SNS topic created
* A phone number subscribed to that topic
* Necessary dependencies in your project

Would you like me to walk you through the **actual Spring Boot code** to send the SMS next?