## **1. Build & Package Your Java Lambda**

Choose one of these packaging methods:

### **📦 A. ZIP or JAR**

* Compile your Java project (with dependencies) using Maven/Gradle.
* Create a deployment artifact:  
  + **Maven**: use **Maven Shade plugin** to produce a fat JAR.
  + **Gradle**: produce a ZIP with the compiled classes and dependencies.

Upload using the AWS Console or CLI:  
  
 bash  
CopyEdit  
aws lambda update-function-code \

--function-name YourFunctionName \

--zip-file fileb://target/your-function.jar \

--region us-east-1

* [github.com+14docs.aws.amazon.com+14youtube.com+14](https://docs.aws.amazon.com/lambda/latest/dg/java-package.html?utm_source=chatgpt.com)

### **🚀 B. Use AWS SAM or CDK**

Define function in a CloudFormation template (SAM) or CDK:  
  
 yaml  
CopyEdit  
Resources:

MyFunction:

Type: AWS::Serverless::Function

Properties:

Handler: com.example.MyHandler::handleRequest

Runtime: java11

CodeUri: target/my-lambda.jar

Environment:

Variables:

USE\_AWS\_SSM: "true"

AWS\_ENABLED: "true"

AWS\_REGION: "us-east-1"

Deploy it:  
  
 bash  
CopyEdit  
sam build

sam deploy --region us-east-1 --stack-name my-stack

or  
  
 bash  
CopyEdit  
cdk synth

cdk deploy

* [blog.tericcabrel.com](https://blog.tericcabrel.com/aws-lambda-java-cdk/?utm_source=chatgpt.com)

## **2. Configure Environment Variables**

You're already using:

ini

CopyEdit

USE\_AWS\_SSM=true

AWS\_ENABLED=true

AWS\_REGION=us-east-1

Make sure to add these in:

* AWS Console → Configuration → Environment variables

Or via CLI:  
  
 bash  
CopyEdit  
aws lambda update-function-configuration \

--function-name YourFunctionName \

--environment "Variables={USE\_AWS\_SSM=true,AWS\_ENABLED=true,AWS\_REGION=us-east-1}"

* [postman.com+6medium.com+6docs.aws.amazon.com+6](https://medium.com/%40tiokachiu/leveraging-aws-lambda-and-postman-for-serverless-api-development-7fabc1d56c1e?utm_source=chatgpt.com)[github.com+1aditi22aggarwal-23582.medium.com+1](https://github.com/travis-ci/dpl/issues/544?utm_source=chatgpt.com)[youtube.com+1postman.com+1](https://www.youtube.com/watch?pp=ygULI2F3c2FuZGphdmE%3D&v=A1rYiHTy9Lg&utm_source=chatgpt.com)

## **3. Create & Set Up API Gateway**

Make the Lambda function accessible over HTTP:

1. In the **API Gateway console**, create a **REST API**.
2. Define a resource (e.g. /yourpath).
3. Add a HTTP method (e.g. POST) and choose **Lambda Proxy Integration**, pointing to your function.
4. Deploy to a stage (e.g. dev) → you'll get an invoke URL.  
    [aditi22aggarwal-23582.medium.com+1blog.tericcabrel.com+1](https://aditi22aggarwal-23582.medium.com/deploy-aws-lambda-as-a-rest-api-java-bee07b800edc?source=post_internal_links---------3----------------------------&utm_source=chatgpt.com)[docs.aws.amazon.com+1youtube.com+1](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-create-api-as-simple-proxy-for-lambda.html?utm_source=chatgpt.com)

## **4. Invoke via Postman**

### **A. Set the request in Postman:**

* **Method**: POST
* **URL**: https://{api-id}.execute-api.us-east-1.amazonaws.com/dev/yourpath
* **Headers**: Content-Type: application/json

**Body**:  
  
 json  
CopyEdit  
{

"someKey": "someValue"

}

### **B. (If required) AWS Auth**

* If your API is private or secured, configure **AWS Signature Authorization** in Postman using your IAM credentials.  
   Its Lambda API support is also built-in for the AWS-provided endpoints.  
   [youtube.com+15apievangelist.com+15docs.aws.amazon.com+15](https://apievangelist.com/2020/04/20/running-and-organizing-aws-lambdas-with-postman-collections/?utm_source=chatgpt.com)[docs.aws.amazon.com](https://docs.aws.amazon.com/lambda/latest/dg/java-package.html?utm_source=chatgpt.com)

### **C. Collection Organisation**

* Save this as part of a Postman Collection.
* You can parameterize variables like stage, region, or function name for easy management.  
   [postman.com+1javatodev.com+1](https://www.postman.com/postman/postman-open-technologies-environments/documentation/sk8qozb/add-environment-variables?utm_source=chatgpt.com)

## **5. (Optional) Use AWS SSM / Secrets Manager**

Since you have USE\_AWS\_SSM=true, if your function expects to fetch secrets or parameters from SSM:

* Grant IAM permission (ssm:GetParameter\*) in your Lambda role.

Your Java code could do:  
  
 java  
CopyEdit  
AWSSimpleSystemsManagement ssm = AWSSimpleSystemsManagementClientBuilder.standard()

.withRegion(System.getenv("AWS\_REGION"))

.build();

GetParameterRequest req = new GetParameterRequest().withName("/my/param").withWithDecryption(true);

String val = ssm.getParameter(req).getParameter().getValue();

* Now deploying this code via Lambda will pull the config dynamically.