# **🚀 Lambda-Ready Jakarta Java Microservices**

A suite of independent microservices using **plain Java (Jakarta EE)** and **Gradle**, each deployable as AWS Lambda functions—**without** Spring Boot or Maven.

## **📦 Services**

1. **api-gateway**
2. **auth-register-service**
3. **auth-login-service**
4. **capinfo-service**
5. **kyc-faceio**
6. **kyc-fingerprint**
7. **key-melissa**
8. **kyc-veriff**
9. **Kyc-sanctions**

Each service is independent and follows the same deployment pattern.

## **🧰 Prerequisites**

* **JDK 17+** (set JAVA\_HOME)
* **Gradle Wrapper** (./gradlew)
* **AWS CLI v2** (configured)
* **Docker & SAM CLI** (optional, for local Lambda emulation)

## **🛠 Build & Run Commands**

Each service can be built, packaged, and run locally—along these lines:

| **Service Name** | **Build Command** | **Run Locally (JVM)** |
| --- | --- | --- |
| **api-gateway** | ./gradlew :api-gateway:clean :api-gateway:build --refresh-dependencies | java -cp "api-gateway/build/libs/api-gateway-1.0.jar:api-gateway/build/libs/lib/\*" com.fundfy.apigateway.Main |
| **auth-register-service** | ./gradlew :auth-register-service:clean :auth-register-service:build --refresh-dependencies | java -cp "auth-register-service/build/libs/auth-register-service-1.0.jar:auth-register-service/build/libs/lib/\*" com.fundfy.authregister.Main |
| **auth-login-service** | ./gradlew :auth-login-service:clean :auth-login-service:build --refresh-dependencies | java -cp "auth-login-service/build/libs/auth-login-service-1.0.jar:auth-login-service/build/libs/lib/\*" com.fundfy.authlogin.Main |
| **capinfo-service** | ./gradlew :capinfo-service:clean: capinfo-service:build --refresh-dependencies | java -cp "capinfo-service/build/libs/capinfo-service-1.0.jar:capinfo-service/build/libs/lib/\*" com.fundfy.capinfo.Main |
| **kyc-faceio** | ./gradlew :kyc-faceio:clean :kyc-faceio:build --refresh-dependencies | java -cp "kyc-faceio/build/libs/kyc-faceio-1.0.jar:kyc-faceio/build/libs/lib/\*" com.fundfy.kycfaceio.Main |
| **kyc-fingerprint** | ./gradlew :kyc-fingerprint:clean :kyc-fingerprint:build --refresh-dependencies | java -cp "kyc-fingerprint/build/libs/kyc-fingerprint-1.0.jar:kyc-fingerprint/build/libs/lib/\*" com.fundfy.kycfingerprint.Main |
| **key-melissa** | ./gradlew :key-melissa:clean :key-melissa:build --refresh-dependencies | java -cp "key-melissa/build/libs/key-melissa-1.0.jar:key-melissa/build/libs/lib/\*" com.fundfy.keymelissa.Main |
| **kyc-veriff** | ./gradlew :kyc-veriff:clean :kyc-veriff:build --refresh-dependencies | java -cp "kyc-veriff/build/libs/kyc-veriff-1.0.jar:kyc-veriff/build/libs/lib/\*" com.fundfy.kycveriff.Main |
| **kyc-sanctions** | ./gradlew :kyc-sanctions:clean :kyc-sanctions:build --refresh-dependencies | java -cp "kyc-sanctions/build/libs/kyc-sanctions-1.0.jar:kyc-sanctions/build/libs/lib/\*" com.fundfy.kycsanctions.Main |

* :clean removes previous build artifacts.
* :build compiles, tests, and creates JAR artifacts.
* --refresh-dependencies ensures Gradle checks and retrieves updated versions ([quarkus.io](https://quarkus.io/guides/aws-lambda?utm_source=chatgpt.com), [stackoverflow.com](https://stackoverflow.com/questions/13565082/how-can-i-force-gradle-to-redownload-dependencies?utm_source=chatgpt.com)).
* The java -cp command ensures both the service JAR and its dependencies in lib/ are on the classpath.

## **🎁 AWS Lambda Packaging**

Each service includes a buildZip task that packages artifacts into:

build/distributions/<service-name>.zip

Add this to build.gradle:

task buildZip(type: Zip) {

from compileJava

from processResources

into('lib') {

from configurations.runtimeClasspath

}

}

build.dependsOn buildZip

This structure complies with AWS guidelines for Lambda Java deployment (dependencies in lib/) ([medium.com](https://medium.com/i-love-my-local-farmer-engineering-blog/choosing-a-packaging-mechanism-for-java-based-aws-lambda-functions-2f006c72c605?utm_source=chatgpt.com), [github.com](https://github.com/awslabs/aws-serverless-java-container/blob/main/samples/spring/pet-store/build.gradle?utm_source=chatgpt.com), [quarkus.io](https://quarkus.io/guides/aws-lambda?utm_source=chatgpt.com)).

Deploy using AWS CLI:

aws lambda update-function-code \

--function-name api-gateway \

--zip-file fileb://api-gateway/build/distributions/api-gateway.zip

## **🔄 Local Lambda Emulation**

Use AWS SAM CLI for local testing:

sam build

sam local invoke api-gateway --event events/gateway-event.json

sam local start-api

Then access via:

http://localhost:3000/<endpoint>

## **🎯 Summary**

This README.md provides a standardized lifecycle across your service ecosystem:

1. **Build** with dependency refresh
2. **Run** locally using direct Java classpath
3. **Package** into Lambda-ready ZIP
4. **Deploy** via AWS CLI or SAM