1. **To view keyboard shortcuts, press question mark**
2. **View keyboard shortcuts**

Grok 3

See new posts

ти будеш допомагати мені з заспуском телеграм-бота на java. Я тобі надсилаю файл build.gradle, щоб ти розумів, з якими бібліотеками я працюю. plugins { id 'java' }group = 'co.biletskyi' version = '1.0-SNAPSHOT' sourceCompatibility = '11'repositories { mavenCentral() }ext { set('lombokVersion', '1.18.16') set('log4jVersion', '2.13.3') set('jacksonVersion', '2.11.3') }dependencies { implementation "org.apache.logging.log4j:log4j-core:${log4jVersion}" implementation "org.apache.logging.log4j:log4j-api:${log4jVersion}" implementation "org.apache.logging.log4j:log4j-slf4j18-impl:${log4jVersion}" implementation 'com.amazonaws:aws-lambda-java-log4j2:1.2.0' implementation "com.fasterxml.jackson.core:jackson-core:${jacksonVersion}" implementation "com.fasterxml.jackson.core:jackson-annotations:${jacksonVersion}" implementation "com.fasterxml.jackson.core:jackson-databind:${jacksonVersion}"

compileOnly "org.projectlombok:lombok:${lombokVersion}"

annotationProcessor "org.projectlombok:lombok:${lombokVersion}"

}task buildZip(type: Zip) { from compileJava from processResources into('lib') { from configurations.runtimeClasspath } }build.dependsOn buildZip

Дякую за надісланий файл build.gradle. Я бачу, що ти використовуєш Java 11, Lombok для спрощення коду, Log4j для логування, Jackson для роботи з JSON, а також налаштування для AWS Lambda. Твоя мета — створити Telegram-бота на Java, але в build.gradle я не бачу бібліотеки для роботи з Telegram API. Для початку давай додамо потрібну залежність і розглянемо базову структуру бота.

1. Додавання бібліотеки для Telegram-бота

Для роботи з Telegram Bot API рекомендую використати бібліотеку Java Telegram Bot API (org.telegram:telegrambots). Додай наступну залежність до build.gradle:

gradle

dependencies {

implementation 'org.telegram:telegrambots:6.9.0' // Актуальна версія на момент травня 2025

}

Після цього онови проєкт, щоб Gradle підтягнув бібліотеку:

bash

./gradlew build

2. Базова структура Telegram-бота

Ось приклад простого Telegram-бота, який відповідає на команду /start. Створи клас, наприклад, TelegramBot:

java

import org.telegram.telegrambots.bots.TelegramLongPollingBot;

import org.telegram.telegrambots.meta.api.methods.send.SendMessage;

import org.telegram.telegrambots.meta.api.objects.Update;

import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

public class TelegramBot extends TelegramLongPollingBot {

private final String botUsername;

private final String botToken;

public TelegramBot(String botUsername, String botToken) {

this.botUsername = botUsername;

this.botToken = botToken;

}

@Override

public String getBotUsername() {

return botUsername;

}

@Override

public String getBotToken() {

return botToken;

}

@Override

public void onUpdateReceived(Update update) {

if (update.hasMessage() && update.getMessage().hasText()) {

String messageText = update.getMessage().getText();

long chatId = update.getMessage().getChatId();

if (messageText.equals("/start")) {

SendMessage message = new SendMessage();

message.setChatId(String.valueOf(chatId));

message.setText("Привіт! Я твій бот. Напиши щось, щоб продовжити.");

try {

execute(message);

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

}

}

}

3. Запуск бота

Щоб запустити бот, потрібно створити головний клас, який ініціалізує бот і реєструє його в Telegram. Наприклад:

java

import org.telegram.telegrambots.meta.TelegramBotsApi;

import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;

public class Main {

public static void main(String[] args) {

try {

TelegramBotsApi botsApi = new TelegramBotsApi(DefaultBotSession.class);

botsApi.registerBot(new TelegramBot("@YourBotUsername", "YOUR\_BOT\_TOKEN"));

System.out.println("Бот успішно запущено!");

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

}

* Заміни @YourBotUsername на ім’я твого бота (починається з @).
* Заміни YOUR\_BOT\_TOKEN на токен, отриманий від BotFather у Telegram.

4. Отримання токена для бота

* Відкрий Telegram і напиши

@BotFather

.

* Відправ команду /start.
* Виконай команду /newbot, щоб створити нового бота.
* Дотримуйся інструкцій: задай ім’я та username бота (username має закінчуватися на Bot, наприклад, @MyTestBot).
* BotFather надішле тобі токен, який потрібно використовувати в коді.

5. Налаштування логування

Оскільки ти використовуєш Log4j, додай конфігурацію для логування. Створи файл log4j2.xml у папці src/main/resources:

xml

<?xml version="1.0" encoding="UTF-8"?>

<Configuration status="WARN">

<Appenders>

<Console name="Console" target="SYSTEM\_OUT">

<PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n"/>

</Console>

</Appenders>

<Loggers>

<Root level="info">

<AppenderRef ref="Console"/>

</Root>

</Loggers>

</Configuration>

Тепер ти можеш використовувати Log4j у коді:

java

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

public class TelegramBot extends TelegramLongPollingBot {

private static final Logger logger = LogManager.getLogger(TelegramBot.class);

// ...

@Override

public void onUpdateReceived(Update update) {

logger.info("Отримано повідомлення: {}", update.getMessage().getText());

// решта коду

}

}

6. Деплой на AWS Lambda (опціонально)

Оскільки в build.gradle є залежність для AWS Lambda, я припускаю, що ти можеш планувати деплой бота на Lambda. Telegram-боти, які використовують Long Polling (як у прикладі вище), не дуже підходять для Lambda, оскільки Lambda має обмеження на тривалість виконання. Для Lambda краще використовувати Webhook:

* Налаштуй Webhook у Telegram:
  + Відправ запит до https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=<YOUR\_LAMBDA\_ENDPOINT>.
* Створи Lambda-функцію, яка обробляє HTTP-запити від Telegram.
* Використовуй Jackson для парсингу JSON-даних від Telegram.

Якщо потрібен приклад для Webhook і Lambda, дай знати, я розпишу детальніше.

7. Наступні кроки

* Тестування локально: Запусти Main клас, щоб перевірити, чи бот відповідає на /start.
* Додавання функціоналу: Напиши, які команди чи функції ти хочеш реалізувати (наприклад, відправка картинок, обробка кнопок, інтеграція з API тощо), і я допоможу з кодом.
* Питання по AWS: Якщо плануєш деплой на Lambda, уточни, чи потрібна допомога з налаштуванням Webhook чи інфраструктури.

Якщо є конкретні питання чи потрібен код для певної функції, напиши, і я допоможу!

допиши мені усі необхідні класи, щоб я міг написати телеграм боту "привіт", а він мені відповідав те сема слово "привіт". Або я пишу: "дерево", і він відповідає "дерево". Бот буде дійсно запускатися на AWS Lambda

Для створення Telegram-бота, який відповідає на будь-яке повідомлення тим самим текстом (наприклад, на "привіт" відповідає "привіт", на "дерево" — "дерево") і розгортається на AWS Lambda, нам потрібно налаштувати Webhook, оскільки Lambda не підтримує Long Polling через обмеження на тривалість виконання. Я пропишу всі необхідні класи, конфігурації та поясню, як це працює.

Огляд рішення

* Бот: Використовуємо бібліотеку telegrambots для обробки повідомлень.
* Webhook: Бот отримує оновлення через HTTP-запити від Telegram.
* AWS Lambda: Обробляє запити через API Gateway, парсить JSON від Telegram і відправляє відповідь.
* Jackson: Для парсингу JSON у Lambda.
* Log4j: Для логування.
* Lombok: Для спрощення коду.

1. Оновлення build.gradle

Твій build.gradle уже містить потрібні залежності для Jackson, Log4j і Lombok, але додамо бібліотеку для Telegram і AWS Lambda. Ось повний файл із додаванням telegrambots:

gradle

plugins {

id 'java'

}

group = 'co.biletskyi'

version = '1.0-SNAPSHOT'

sourceCompatibility = '11'

repositories {

mavenCentral()

}

ext {

set('lombokVersion', '1.18.16')

set('log4jVersion', '2.13.3')

set('jacksonVersion', '2.11.3')

set('telegramVersion', '6.9.0')

}

dependencies {

implementation "org.apache.logging.log4j:log4j-core:${log4jVersion}"

implementation "org.apache.logging.log4j:log4j-api:${log4jVersion}"

implementation "org.apache.logging.log4j:log4j-slf4j18-impl:${log4jVersion}"

implementation 'com.amazonaws:aws-lambda-java-log4j2:1.2.0'

implementation "com.fasterxml.jackson.core:jackson-core:${jacksonVersion}"

implementation "com.fasterxml.jackson.core:jackson-annotations:${jacksonVersion}"

implementation "com.fasterxml.jackson.core:jackson-databind:${jacksonVersion}"

implementation "org.telegram:telegrambots:${telegramVersion}"

implementation 'com.amazonaws:aws-lambda-java-core:1.2.1'

implementation 'com.amazonaws:aws-lambda-java-events:3.11.0'

compileOnly "org.projectlombok:lombok:${lombokVersion}"

annotationProcessor "org.projectlombok:lombok:${lombokVersion}"

}

task buildZip(type: Zip) {

from compileJava

from processResources

into('lib') {

from configurations.runtimeClasspath

}

}

build.dependsOn buildZip

Онови проєкт:

bash

./gradlew build

2. Конфігурація Log4j

Створи файл src/main/resources/log4j2.xml для логування:

xml

<?xml version="1.0" encoding="UTF-8"?>

<Configuration status="WARN">

<Appenders>

<Console name="Console" target="SYSTEM\_OUT">

<PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n"/>

</Console>

</Appenders>

<Loggers>

<Root level="info">

<AppenderRef ref="Console"/>

</Root>

</Loggers>

</Configuration>

3. Код бота

Нижче наведено всі необхідні класи для реалізації бота.

3.1. Клас TelegramBot

Цей клас відповідає за логіку бота: обробляє оновлення від Telegram і відправляє відповідь.

java

package co.biletskyi.telegram;

import com.fasterxml.jackson.databind.ObjectMapper;

import lombok.Getter;

import lombok.Setter;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.telegram.telegrambots.bots.TelegramWebhookBot;

import org.telegram.telegrambots.meta.api.methods.BotApiMethod;

import org.telegram.telegrambots.meta.api.methods.send.SendMessage;

import org.telegram.telegrambots.meta.api.objects.Update;

@Getter

@Setter

public class TelegramBot extends TelegramWebhookBot {

private static final Logger logger = LogManager.getLogger(TelegramBot.class);

private String botUsername;

private String botToken;

private String botPath;

public TelegramBot(String botUsername, String botToken, String botPath) {

this.botUsername = botUsername;

this.botToken = botToken;

this.botPath = botPath;

}

@Override

public BotApiMethod<?> onWebhookUpdateReceived(Update update) {

if (update.hasMessage() && update.getMessage().hasText()) {

String receivedText = update.getMessage().getText();

long chatId = update.getMessage().getChatId();

logger.info("Отримано повідомлення: {}", receivedText);

SendMessage message = new SendMessage();

message.setChatId(String.valueOf(chatId));

message.setText(receivedText); // Відповідаємо тим самим текстом

return message;

}

return null;

}

@Override

public String getBotPath() {

return botPath;

}

}

3.2. Клас LambdaHandler

Цей клас обробляє запити від AWS Lambda, парсить JSON і передає оновлення боту.

java

package co.biletskyi.telegram;

import com.amazonaws.services.lambda.runtime.Context;

import com.amazonaws.services.lambda.runtime.RequestStreamHandler;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.telegram.telegrambots.meta.api.objects.Update;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.nio.charset.StandardCharsets;

public class LambdaHandler implements RequestStreamHandler {

private static final Logger logger = LogManager.getLogger(LambdaHandler.class);

private static final ObjectMapper objectMapper = new ObjectMapper();

private static final TelegramBot bot = new TelegramBot(

"@YourBotUsername", // Заміни на свій username

"YOUR\_BOT\_TOKEN", // Заміни на свій токен

"/webhook" // Шлях для Webhook

);

@Override

public void handleRequest(InputStream inputStream, OutputStream outputStream, Context context) throws IOException {

try {

// Читаємо вхідний JSON

String json = new String(inputStream.readAllBytes(), StandardCharsets.UTF\_8);

logger.info("Отримано запит: {}", json);

// Парсимо Update

Update update = objectMapper.readValue(json, Update.class);

// Обробляємо оновлення

var response = bot.onWebhookUpdateReceived(update);

// Відправляємо відповідь (якщо є)

if (response != null) {

String responseJson = objectMapper.writeValueAsString(response);

outputStream.write(responseJson.getBytes(StandardCharsets.UTF\_8));

} else {

outputStream.write("{\"status\":\"ok\"}".getBytes(StandardCharsets.UTF\_8));

}

} catch (Exception e) {

logger.error("Помилка обробки запиту", e);

outputStream.write("{\"error\":\"Internal Server Error\"}".getBytes(StandardCharsets.UTF\_8));

}

}

}

4. Налаштування Webhook

Після деплою на AWS Lambda тобі потрібно налаштувати Webhook у Telegram. Виконай HTTP-запит (наприклад, через curl або Postman):

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=<YOUR\_API\_GATEWAY\_URL>/webhook"

* <YOUR\_BOT\_TOKEN> — токен від BotFather.
* <YOUR\_API\_GATEWAY\_URL> — URL твого API Gateway, створеного для Lambda (наприклад, https://abc123.execute-api.us-east-1.amazonaws.com/prod).

Переконайся, що шлях /webhook відповідає botPath у TelegramBot.

5. Деплой на AWS Lambda

* Скомпілюй проєкт:

bash

./gradlew buildZip

Це створить ZIP-архів у build/distributions.

* Створи Lambda-функцію:
  + У консолі AWS перейди до Lambda.
  + Створи нову функцію, вибери Java 11 як runtime.
  + Завантаж ZIP-архів із build/distributions.
  + Вкажи Handler: co.biletskyi.telegram.LambdaHandler::handleRequest.
* Налаштуй API Gateway:
  + Створи API Gateway у консолі AWS.
  + Додай POST-метод із шляхом /webhook, який перенаправляє запити на твою Lambda-функцію.
  + Розгорни API, щоб отримати URL (наприклад, https://abc123.execute-api.us-east-1.amazonaws.com/prod).
* Налаштуй Webhook: Виконай запит із пункту 4, щоб Telegram відправляв оновлення на твій API Gateway.

6. Тестування

* Напиши боту в Telegram будь-яке повідомлення, наприклад, "привіт" або "дерево".
* Бот має відповісти тим самим текстом.
* Переглянь логи в AWS CloudWatch, щоб перевірити, чи немає помилок.

7. Замітки

* Токен і Username: Заміни @YourBotUsername і YOUR\_BOT\_TOKEN у LambdaHandler на реальні значення від BotFather.
* Безпека: Зберігай токен у AWS Secrets Manager або Environment Variables замість хардкоду в коді.
* Ліміт Lambda: Переконайся, що функція завершується швидко (менше 15 хвилин), інакше Telegram може не отримати відповідь.
* Розширення: Якщо хочеш додати нові функції (наприклад, команди чи кнопки), напиши, і я допоможу з кодом.

Якщо потрібні додаткові пояснення чи допомога з налаштуванням AWS, дай знати!

Добре. Дай відповідь на наступне питання. У класі LambdaHandler в коді: private static final TelegramBot bot = new TelegramBot( "@YourBotUsername", // Заміни на свій username "YOUR\_BOT\_TOKEN", // Заміни на свій токен "/webhook" // Шлях для Webhook );є шлях для вебхука. Але я не розумію, яким він має бути, тобто що саме треба написати в цьому рядку кода? Invoke URL від API Gateway , який мені дає AWS? Чи HTTP запит? Поясни цей момент

Дякую за уточнення! Давай розберемо, що саме означає botPath (шлях для Webhook) у рядку:

java

private static final TelegramBot bot = new TelegramBot(

"@YourBotUsername", // Заміни на свій username

"YOUR\_BOT\_TOKEN", // Заміни на свій токен

"/webhook" // Шлях для Webhook

);

і як він пов’язаний із AWS API Gateway та Telegram Webhook.

Що таке botPath?

Параметр botPath у класі TelegramBot (який розширює TelegramWebhookBot) визначає кінцеву частину URL, на яку Telegram буде відправляти оновлення (повідомлення від користувачів). Цей шлях є частиною повного URL, який ти реєструєш у Telegram через запит setWebhook. Повний URL виглядатиме так:

https://<YOUR\_API\_GATEWAY\_URL>/<botPath>

* <YOUR\_API\_GATEWAY\_URL> — це базовий URL, який ти отримуєш від AWS API Gateway після створення та розгортання API (наприклад, https://abc123.execute-api.us-east-1.amazonaws.com/prod).
* <botPath> — це додаткова частина шляху, яка вказує Telegram, куди саме відправляти запити. У твоєму коді це /webhook.

Що написати в botPath?

У рядку коду:

java

"/webhook" // Шлях для Webhook

значення /webhook є шляхом, який ти визначаєш для обробки запитів у API Gateway. Це не повний URL, а лише частина, яка додається до базового URL API Gateway. Ти можеш залишити /webhook або вибрати будь-який інший шлях, наприклад, /telegram або /bot, головне, щоб:

* Цей шлях відповідав налаштуванням у API Gateway.
* Ти вказав цей шлях у запиті setWebhook для Telegram.

Як це працює з API Gateway?

* Налаштування API Gateway:
  + У консолі AWS ти створюєш API Gateway і додаєш ресурс із шляхом, наприклад, /webhook.
  + Цей ресурс пов’язується з твоєю Lambda-функцією (LambdaHandler).
  + Після розгортання API Gateway ти отримуєш Invoke URL, наприклад:

https://abc123.execute-api.us-east-1.amazonaws.com/prod

* + Повний URL для Webhook буде:

https://abc123.execute-api.us-east-1.amazonaws.com/prod/webhook

* Реєстрація Webhook у Telegram:
  + Ти відправляєш запит до Telegram API, щоб зареєструвати Webhook:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://abc123.execute-api.us-east-1.amazonaws.com/prod/webhook"

* + Тут /webhook у кінці URL відповідає значенню botPath у коді.
* Обробка в коді:
  + У класі TelegramBot метод getBotPath() повертає /webhook.
  + Бібліотека telegrambots використовує цей шлях, щоб зрозуміти, які запити належать до цього бота.
  + У LambdaHandler вхідні запити (JSON від Telegram) надходять через API Gateway, і код парсить їх, передаючи в TelegramBot.

Чи це Invoke URL від API Gateway?

Ні, botPath — це не повний Invoke URL, а лише частина шляху, яка додається до Invoke URL. Invoke URL — це базова частина, яку ти отримуєш від API Gateway (наприклад, https://abc123.execute-api.us-east-1.amazonaws.com/prod). У коді ти вказує51ш лише /webhook як botPath, а повний URL формується, коли ти реєструєш Webhook у Telegram.

Чи це HTTP-запит?

Ні, botPath — це не HTTP-запит, а просто рядок, який визначає шлях у URL. HTTP-запит потрібен окремо, щоб зареєструвати Webhook у Telegram (див. приклад із curl вище).

Що саме писати в коді?

У твоєму випадку залиш /webhook у коді, якщо ти налаштував API Gateway із ресурсом /webhook. Наприклад:

java

private static final TelegramBot bot = new TelegramBot(

"@YourBotUsername",

"YOUR\_BOT\_TOKEN",

"/webhook" // Це шлях, який додається до Invoke URL

);

Якщо ти хочеш використати інший шлях, наприклад, /telegram, то:

* Зміни в коді:

java

"/telegram"

* Налаштуй ресурс /telegram в API Gateway.
* Зареєструй Webhook із новим шляхом:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://abc123.execute-api.us-east-1.amazonaws.com/prod/telegram"

Покроковий приклад

* Ти отримав Invoke URL від API Gateway:

https://abc123.execute-api.us-east-1.amazonaws.com/prod

* У API Gateway створив ресурс із шляхом /webhook.
* У коді залишив botPath як /webhook.
* Зареєстрував Webhook у Telegram:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://abc123.execute-api.us-east-1.amazonaws.com/prod/webhook"

* Telegram відправляє оновлення на цей URL, і твоя Lambda-функція їх обробляє.

Що перевірити?

* API Gateway: Переконайся, що ресурс (наприклад, /webhook) створено і пов’язано з Lambda.
* Invoke URL: Скопіюй Invoke URL із консолі AWS і додай до нього /webhook для запиту setWebhook.
* Логи: Якщо Webhook не працює, перевір логи в CloudWatch, щоб побачити, чи доходять запити до Lambda.

Якщо щось незрозуміло

* Якщо не знаєш, як знайти Invoke URL, напиши, і я поясню, де його взяти в консолі AWS.
* Якщо потрібна допомога з налаштуванням API Gateway чи реєстрацією Webhook, уточни, і я розпишу детальніше.

Сподіваюся, це прояснило ситуацію! Дай знати, якщо є інші питання.

Деталі налаштування API Gateway (дай відповідь коротко та коректно) ресурс /webhook створено. Питання таке: який метод прописати у цьому ресурсі ( ресурс те саме що route)? Я думаю залишити POST

Так, ресурс у API Gateway — це те саме, що route. Для ресурсу /webhook використовуй метод POST, оскільки Telegram відправляє оновлення через POST-запити.

Коротке налаштування:

* У консолі AWS API Gateway вибери свій API.
* У розділі Resources знайди створений ресурс /webhook.
* Додай метод: POST.
* У налаштуваннях метода:
  + Integration Type: Lambda Function.
  + Вкажи свою Lambda-функцію (наприклад, LambdaHandler).
  + Увімкни Use Lambda Proxy Integration (для передачі JSON від Telegram).
* Розгорни API (Deploy API) у потрібній stage (наприклад, prod).
* Скопіюй Invoke URL (наприклад, https://abc123.execute-api.us-east-1.amazonaws.com/prod).
* Зареєструй Webhook у Telegram:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://abc123.execute-api.us-east-1.amazonaws.com/prod/webhook"

Чому POST?

Telegram Webhook відправляє JSON із оновленнями через HTTP POST, тому GET чи інші методи не підходять.

Якщо потрібні додаткові деталі, напиши!

Ще одне коротке запитання , я хочу коротку та коректну відповідь. в API Gateway налаштуваннях я обрав вкладку Develop->Integrations. Бачу налаштування "Manage Integrations". Чи потрібно в моєму випадку налаштовувати Parameter mapping ?

Ні, у твоєму випадку не потрібно налаштовувати Parameter Mapping у "Manage Integrations". Оскільки ти використовуєш Lambda Proxy Integration для ресурсу /webhook із методом POST, API Gateway автоматично передає весь вхідний JSON-запит (від Telegram) у Lambda без додаткового мапінгу.

Потрібні твої підказки щодо деяких кроків у AWS. Відповідай коротко , коректно та по суті.

* У мене вже є створена Lambda функція та вказаний Handler.
* ZIP-архів також уже завантажений в AWS Lambda.
* У мене є створений API Gateway, в якому створений Stage "stage".
* Також, в моєму API Gateway вказаний route "/webhook" з методом POST.

Питання наступні: Питання №1. Коли я створював API Gateway , то вказав у налаштуваннях Protocol "HTTP". Наразі це також видно на сторінці AWS Console. Чи це коректно в моєму випадку? Питання №2. В налаштуваннях Lambda-функції в AWS Console є вкладка Triggers. Вона пуста наразі. Розкажи коротко та по суті , як мені це виправити

Питання №1: Чи коректно використовувати HTTP-протокол для API Gateway?

Так, протокол HTTP у API Gateway коректний для твого випадку. Telegram Webhook працює через HTTP (HTTPS, точніше, оскільки AWS забезпечує SSL). HTTP API Gateway дешевший і достатній для обробки POST-запитів від Telegram. Переконайся, що твій Webhook URL (Invoke URL + /webhook) використовує https://.

Питання №2: Як виправити пусту вкладку Triggers у Lambda?

Щоб додати API Gateway як тригер для Lambda:

* Перейди до AWS Console -> Lambda -> твоя функція.
* У вкладці Triggers натисни Add Trigger.
* Вибери API Gateway.
* Вкажи:
  + API: Твій створений API Gateway.
  + Stage: stage (або той, що ти створив).
  + Resource: /webhook, метод POST.
* Увімкни Lambda Proxy Integration (якщо ще не ввімкнено).
* Натисни Add і збережи.

Це зв’яже API Gateway із Lambda, і запити на /webhook будуть викликати твою функцію. Переконайся, що Invoke URL із API Gateway зареєстровано у Telegram через setWebhook.

Питання №3 (так само коротко та по суті): коли мені натискати кнопку Deploy в APi Gateway?

Натискай кнопку Deploy у API Gateway після будь-яких змін у конфігурації API (наприклад, додавання/зміна ресурсу /webhook, метода POST, інтеграції з Lambda чи оновлення Stage).

Коли саме:

* Після створення чи оновлення ресурсу /webhook і налаштування метода POST.
* Після зв’язування API Gateway із Lambda у налаштуваннях інтеграції.
* Вибери Stage (наприклад, stage) і розгорни, щоб зміни набули чинності.

Це оновить Invoke URL (наприклад, https://abc123.execute-api.us-east-1.amazonaws.com/stage), який ти використовуєш у setWebhook для Telegram.

Що я спостерігаю:

* Після додавання тригера бачу новий API Endpoint:

API endpoint: https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/TelegramBotLambdaFunctionТакож, у налаштуваннях API Gateway автоматично додався новий ресурс (route). Тепер він має шлях /TelegramBotLambdaFunctionЧи це коректно? Поясни коротко, чому так

Чи коректно, що з’явився новий ресурс /TelegramBotLambdaFunction?

Ні, це не зовсім коректно для твого випадку, якщо ти плануєш використовувати ресурс /webhook для Telegram Webhook. Новий ресурс /TelegramBotLambdaFunction і відповідний API Endpoint (https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/TelegramBotLambdaFunction) з’явилися, тому що ти додав API Gateway як тригер у Lambda, і AWS автоматично створив новий ресурс, названий за ім’ям Lambda-функції.

Чому так сталося?

* Коли ти додаєш API Gateway як тригер у вкладці Triggers у Lambda, AWS автоматично створює новий ресурс у твоєму API Gateway із шляхом, який відповідає імені Lambda-функції (у твоєму випадку /TelegramBotLambdaFunction).
* Це стандартна поведінка AWS для спрощення налаштування, але вона не враховує твій існуючий ресурс /webhook, який ти створив для Telegram Webhook.

Що робити?

* Залишайся з /webhook:
  + Telegram Webhook налаштовано для /webhook (згідно з твоїм кодом, де botPath = "/webhook").
  + Новий ресурс /TelegramBotLambdaFunction тобі не потрібен, якщо ти хочеш використовувати /webhook.
* Виправлення:
  + Перейди до API Gateway у консолі AWS.
  + Видали автоматично створений ресурс /TelegramBotLambdaFunction, якщо він не потрібен.
  + Переконайся, що ресурс /webhook із методом POST правильно інтегрований із твоєю Lambda-функцією.
  + У вкладці Integrations для /webhook перевір, що обрано твою Lambda-функцію та увімкнено Lambda Proxy Integration.
* Оновлення тригера:
  + У Lambda, у вкладці Triggers, видали тригер, який посилається на /TelegramBotLambdaFunction.
  + Додай тригер вручну, вказавши існуючий API Gateway і ресурс /webhook (Stage: stage, метод POST).
* Розгортання:
  + Після внесення змін натисни Deploy у API Gateway, вибравши Stage stage.
  + Використовуй правильний Invoke URL для Webhook:

https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/webhook

* + Переконайся, що цей URL зареєстровано в Telegram через:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/webhook"

Чому це важливо?

Telegram очікує, що оновлення надходитимуть на шлях /webhook (згідно з botPath у коді). Якщо використовувати /TelegramBotLambdaFunction, бот не оброблятиме запити, оскільки шлях не збігається.

Якщо потрібні деталі чи допомога з видаленням ресурсу, напиши!

На цьому етапі виникли проблеми, які треба вирішити. Справа у тому, що AWS не дозволяє вказати інший шлях при створенні тригера . Він автоматично прописує /TelegramBotLambdaFunction. Ти впевнений, що це є проблемою?

Дякую за уточнення! Давай розберемо ситуацію: AWS автоматично створює ресурс /TelegramBotLambdaFunction при додаванні тригера, і ти не можеш вказати інший шлях (наприклад, /webhook) безпосередньо в налаштуваннях тригера. Розберу, чи це проблема і як її вирішити.

Чи є автоматичний шлях /TelegramBotLambdaFunction проблемою?

Так, це проблема у твоєму випадку, якщо ти хочеш, щоб Telegram Webhook працював із шляхом /webhook, як вказано в коді (botPath = "/webhook"). Telegram відправляє оновлення на URL, який ти реєструєш через setWebhook, і цей URL має закінчуватися на /webhook (наприклад, https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/webhook). Якщо Telegram відправляє запити на /TelegramBotLambdaFunction, бібліотека telegrambots їх не обробить, оскільки шлях не збігається з botPath.

Чому AWS примусово створює /TelegramBotLambdaFunction?

Коли ти додаєш API Gateway як тригер у Lambda через вкладку Triggers, AWS автоматично:

* Створює новий ресурс у API Gateway із шляхом, що відповідає імені Lambda-функції (/TelegramBotLambdaFunction).
* Налаштовує інтеграцію цього ресурсу з Lambda. Це спрощує швидке налаштування, але не дозволяє вибрати існуючий ресурс (наприклад, /webhook).

Як виправити?

Щоб використовувати /webhook замість /TelegramBotLambdaFunction, виконай такі кроки:

* Видали автоматично створений тригер:
  + У консолі AWS перейди до Lambda -> твоя функція -> вкладка Triggers.
  + Видали тригер, який посилається на /TelegramBotLambdaFunction.
* Налаштуй інтеграцію через API Gateway:
  + Перейди до API Gateway -> твій API.
  + У розділі Resources переконайся, що ресурс /webhook із методом POST уже створено.
  + У налаштуваннях метода POST для /webhook (вкладка Integrations):
    - Вибери Lambda Function як тип інтеграції.
    - Вкажи свою Lambda-функцію (TelegramBotLambdaFunction).
    - Увімкни Use Lambda Proxy Integration.
    - Збережи.
* Додай Lambda до тригерів через API Gateway:
  + У API Gateway, у ресурсі /webhook, після налаштування інтеграції ти вже зв’язав Lambda.
  + У Lambda у вкладці Triggers з’явиться тригер автоматично після збереження інтеграції в API Gateway. Якщо ні:
    - У Lambda натисни Add Trigger -> вибери API Gateway -> вкажи свій API, Stage (stage) і ресурс /webhook (метод POST).
* Розгорни API:
  + У API Gateway натисни Deploy, вибери Stage stage.
  + Скопіюй Invoke URL (наприклад, https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage).
* Онови Webhook у Telegram:
  + Зареєструй правильний URL для Webhook:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/webhook"

* Видали непотрібний ресурс (опціонально):
  + У API Gateway видали ресурс /TelegramBotLambdaFunction, якщо він не потрібен, щоб уникнути плутанини.

Чи можна залишити /TelegramBotLambdaFunction?

Теоретично можна, але тоді потрібно змінити код у TelegramBot:

* Зміни botPath із /webhook на /TelegramBotLambdaFunction:

java

private static final TelegramBot bot = new TelegramBot(

"@YourBotUsername",

"YOUR\_BOT\_TOKEN",

"/TelegramBotLambdaFunction"

);

* Онови Webhook у Telegram:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/TelegramBotLambdaFunction"

Це спрацює, але /webhook — більш стандартний і зрозумілий шлях для Telegram-ботів, тому краще використовувати його.

Рекомендація

Використовуй /webhook, як у початковому коді, щоб уникнути змін у коді. Налаштуй інтеграцію через API Gateway, як описано вище, і видали /TelegramBotLambdaFunction, якщо він не потрібен. Це забезпечить відповідність між кодом, API Gateway і Telegram Webhook.

Перевірка

* Після налаштування перевір логи в CloudWatch, щоб переконатися, що запити від Telegram доходять до Lambda.
* Напиши боту в Telegram, щоб перевірити, чи відповідає він.

Якщо потрібна допомога з конкретним кроком у консолі AWS, напиши, і я розпишу детальніше!

Давай спробуємо залишити /TelegramBotLambdaFunction. В такому випадку, я змінюю path у коді. Видаляю route /webhook.

Добре, залишимо /TelegramBotLambdaFunction як шлях для Webhook і видалимо /webhook. Ось короткі та коректні кроки, щоб усе запрацювало:

1. Зміна коду

У класі LambdaHandler зміни botPath на /TelegramBotLambdaFunction:

java

private static final TelegramBot bot = new TelegramBot(

"@YourBotUsername", // Заміни на свій username

"YOUR\_BOT\_TOKEN", // Заміни на свій токен

"/TelegramBotLambdaFunction" // Новий шлях

);

2. Видалення ресурсу /webhook

* У консолі AWS перейди до API Gateway -> твій API -> Resources.
* Вибери ресурс /webhook і видали його (натисни Actions -> Delete Resource).
* Переконайся, що ресурс /TelegramBotLambdaFunction із методом POST залишений і правильно інтегрований із твоєю Lambda-функцією.

3. Перевірка інтеграції

* У API Gateway -> Resources -> /TelegramBotLambdaFunction -> метод POST:
  + Переконайся, що Integration Type — Lambda Function.
  + Вказано твою Lambda-функцію (TelegramBotLambdaFunction).
  + Увімкнено Use Lambda Proxy Integration.
* Збережи зміни.

4. Розгортання API

* У API Gateway натисни Deploy.
* Вибери Stage stage і розгорни.
* Скопіюй Invoke URL (наприклад, https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage).

5. Оновлення Webhook у Telegram

Зареєструй Webhook із новим шляхом:

bash

curl -X GET "https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/setWebhook?url=https://pewyxlwjth.execute-api.eu-north-1.amazonaws.com/stage/TelegramBotLambdaFunction"

6. Перекомпіляція та завантаження

* Онови код із новим botPath і скомпілюй проєкт:

bash

./gradlew buildZip

* Завантаж новий ZIP-архів у Lambda через консоль AWS.

7. Тестування

* Напиши боту в Telegram (наприклад, "привіт").
* Перевір, чи бот відповідає тим самим текстом.
* Переглянь логи в CloudWatch, щоб переконатися, що запити обробляються.

Примітка

* Переконайся, що @YourBotUsername і YOUR\_BOT\_TOKEN у коді замінено на реальні значення.
* Якщо бот не відповідає, перевір, чи Webhook успішно зареєстровано (виконай https://api.telegram.org/bot<YOUR\_BOT\_TOKEN>/getWebhookInfo).

Якщо виникнуть проблеми, напиши, і я допоможу!

Я пробую скомпілювати проєкт вручну.Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details> Task :compileJava FAILED FAILURE: Build failed with an exception.

* What went wrong: Execution failed for task ':compileJava'.> java.lang.IllegalAccessError: class lombok.javac.apt.LombokProcessor (in unnamed module @0x2ca4e6a3) cannot access class com.sun.tools.javac.processing.JavacProcessingEnvironment (in module jdk.compiler) because module jdk.compiler does not export com.sun.tools.javac.processing to unnamed module @0x2ca4e6a3

Помилка java.lang.IllegalAccessError пов’язана з несумісністю Lombok із Java 11 (або новішою версією) через обмеження модульної системи Java (JPMS). У твоєму build.gradle вказано sourceCompatibility = '11', а версія Lombok (1.18.16) може некоректно працювати з модулем jdk.compiler. Ось коротке та коректне рішення:

Причина

Lombok намагається отримати доступ до внутрішнього API com.sun.tools.javac.processing, яке не експортується модулем jdk.compiler у Java 11+. Потрібно або оновити Lombok, або додати JVM-аргументи для експорту.

Рішення

* Онови Lombok до новішої версії: У build.gradle зміни версію Lombok на актуальну (наприклад, 1.18.34):

gradle

ext {

set('lombokVersion', '1.18.34') // Актуальна версія на травень 2025

}

* Додай JVM-аргументи (альтернатива): Якщо оновлення Lombok не допомагає, додай у build.gradle конфігурацію для Gradle, щоб експортувати потрібний модуль:

gradle

tasks.withType(JavaCompile) {

options.fork = true

options.forkOptions.jvmArgs += ['--add-exports', 'jdk.compiler/com.sun.tools.javac.processing=ALL-UNNAMED']

}

* Очисти та скомпілюй: Виконай:

bash

./gradlew clean build

Рекомендація

* Спробуй спочатку оновити Lombok (крок 1), оскільки новіші версії краще сумісні з Java 11.
* Якщо помилка залишиться, додай JVM-аргументи (крок 2).
* Переконайся, що використовуєш JDK 11 (виконай java -version).

Якщо помилка не зникає, напиши, і я допоможу розібратися!

FAILURE: Build failed with an exception.

* What went wrong: Execution failed for task ':compileJava'.> java.lang.IllegalAccessError: class org.gradle.internal.compiler.java.ClassNameCollector (in unnamed module @0x123325d9) cannot access class com.sun.tools.javac.code.Symbol$TypeSymbol (in module jdk.compiler) because module jdk.compiler does not export com.sun.tools.javac.code to unnamed module @0x123325d9

Помилка java.lang.IllegalAccessError вказує на проблему з доступом Gradle до внутрішнього API Java (com.sun.tools.javac.code) через модульну систему Java 11. Це схоже на попередню проблему з Lombok, але тепер проблема в самому Gradle. Ось коротке та коректне рішення:

Рішення

* Додай JVM-аргументи для Gradle: У build.gradle додай конфігурацію, щоб експортувати потрібний модуль:

gradle

tasks.withType(JavaCompile) {

options.fork = true

options.forkOptions.jvmArgs += [

'--add-exports', 'jdk.compiler/com.sun.tools.javac.code=ALL-UNNAMED',

'--add-exports', 'jdk.compiler/com.sun.tools.javac.processing=ALL-UNNAMED'

]

}

* Онови Gradle (опціонально): Переконайся, що використовуєш актуальну версію Gradle. У файлі gradle/wrapper/gradle-wrapper.properties вкажи:

properties

distributionUrl=https\://services.gradle.org/distributions/gradle-8.10-bin.zip

Онови Gradle:

bash

./gradlew wrapper

* Очисти та скомпілюй: Виконай:

bash

./gradlew clean build

Примітка

* Переконайся, що JDK 11 встановлено коректно (java -version має показати 11).
* Якщо помилка не зникає, спробуй використати JDK 17, змінивши sourceCompatibility = '17' у build.gradle.

Якщо проблема залишається, напиши, і я розберу детальніше!

* What went wrong: Could not compile build file 'C:\repository\telegram-bot-standard\build.gradle'.> startup failed: General error during semantic analysis: Unsupported class file major version 67 java.lang.IllegalArgumentException: Unsupported class file major version 67 at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:196) at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:177) at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:163) at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:284) at org.codehaus.groovy.ast.decompiled.AsmDecompiler.parseClass(AsmDecompiler.java:81) at org.codehaus.groovy.control.ClassNodeResolver.findDecompiled(ClassNodeResolver.java:251) at org.codehaus.groovy.control.ClassNodeResolver.tryAsLoaderClassOrScript(ClassNodeResolver.java:189) at org.codehaus.groovy.control.ClassNodeResolver.findClassNode(ClassNodeResolver.java:169) at org.codehaus.groovy.control.ClassNodeResolver.resolveName(ClassNodeResolver.java:125) at org.codehaus.groovy.ast.decompiled.AsmReferenceResolver.resolveClassNullable(AsmReferenceResolver.java:57) at org.codehaus.groovy.ast.decompiled.AsmReferenceResolver.resolveClass(AsmReferenceResolver.java:44) at org.codehaus.groovy.ast.decompiled.AsmReferenceResolver.resolveNonArrayType(AsmReferenceResolver.java:79) at org.codehaus.groovy.ast.decompiled.AsmReferenceResolver.resolveType(AsmReferenceResolver.java:70) at org.codehaus.groovy.ast.decompiled.MemberSignatureParser.createMethodNode(MemberSignatureParser.java:57) at org.codehaus.groovy.ast.decompiled.DecompiledClassNode$2.get(DecompiledClassNode.java:234) at org.codehaus.groovy.ast.decompiled.DecompiledClassNode$2.get(DecompiledClassNode.java:231) at org.codehaus.groovy.ast.decompiled.DecompiledClassNode.createMethodNode(DecompiledClassNode.java:242) at org.codehaus.groovy.ast.decompiled.DecompiledClassNode.lazyInitMembers(DecompiledClassNode.java:199) at org.codehaus.groovy.ast.decompiled.DecompiledClassNode.getDeclaredMethods(DecompiledClassNode.java:122) at org.codehaus.groovy.ast.ClassNode.getMethods(ClassNode.java:912) at org.codehaus.groovy.ast.ClassNode.tryFindPossibleMethod(ClassNode.java:1280) at org.codehaus.groovy.control.StaticImportVisitor.transformMethodCallExpression(StaticImportVisitor.java:252) at org.codehaus.groovy.control.StaticImportVisitor.transform(StaticImportVisitor.java:113) at org.codehaus.groovy.ast.ClassCodeExpressionTransformer.visitExpressionStatement(ClassCodeExpressionTransformer.java:142) at org.codehaus.groovy.ast.stmt.ExpressionStatement.visit(ExpressionStatement.java:40) at org.codehaus.groovy.ast.ClassCodeVisitorSupport.visitClassCodeContainer(ClassCodeVisitorSupport.java:110) at org.codehaus.groovy.ast.ClassCodeVisitorSupport.visitConstructorOrMethod(ClassCodeVisitorSupport.java:121) at org.codehaus.groovy.ast.ClassCodeExpressionTransformer.visitConstructorOrMethod(ClassCodeExpressionTransformer.java:53) at org.codehaus.groovy.control.StaticImportVisitor.visitConstructorOrMethod(StaticImportVisitor.java:89) at org.codehaus.groovy.ast.ClassCodeVisitorSupport.visitConstructor(ClassCodeVisitorSupport.java:128) at org.codehaus.groovy.ast.ClassNode.visitContents(ClassNode.java:1099) at org.codehaus.groovy.ast.ClassCodeVisitorSupport.visitClass(ClassCodeVisitorSupport.java:54) at org.codehaus.groovy.control.StaticImportVisitor.visitClass(StaticImportVisitor.java:83) at org.codehaus.groovy.control.CompilationUnit$14.call(CompilationUnit.java:708) at org.codehaus.groovy.control.CompilationUnit.applyToPrimaryClassNodes(CompilationUnit.java:1084) at org.codehaus.groovy.control.CompilationUnit.doPhaseOperation(CompilationUnit.java:640) at org.codehaus.groovy.control.CompilationUnit.processPhaseOperations(CompilationUnit.java:618) at org.codehaus.groovy.control.CompilationUnit.compile(CompilationUnit.java:595) at groovy.lang.GroovyClassLoader.doParseClass(GroovyClassLoader.java:401) at groovy.lang.GroovyClassLoader.access$300(GroovyClassLoader.java:89) at groovy.lang.GroovyClassLoader$5.provide(GroovyClassLoader.java:341) at groovy.lang.GroovyClassLoader$5.provide(GroovyClassLoader.java:338) at org.codehaus.groovy.runtime.memoize.ConcurrentCommonCache.getAndPut(ConcurrentCommonCache.java:147) at groovy.lang.GroovyClassLoader.parseClass(GroovyClassLoader.java:336) at org.gradle.groovy.scripts.internal.DefaultScriptCompilationHandler.compileScript(DefaultScriptCompilationHandler.java:139) at org.gradle.groovy.scripts.internal.DefaultScriptCompilationHandler.compileToDir(DefaultScriptCompilationHandler.java:95) at org.gradle.groovy.scripts.internal.BuildOperationBackedScriptCompilationHandler$2.run(BuildOperationBackedScriptCompilationHandler.java:54) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:395) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:387) at org.gradle.internal.operations.DefaultBuildOperationExecutor$1.execute(DefaultBuildOperationExecutor.java:157) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:242) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:150) at org.gradle.internal.operations.DefaultBuildOperationExecutor.run(DefaultBuildOperationExecutor.java:84) at org.gradle.internal.operations.DelegatingBuildOperationExecutor.run(DelegatingBuildOperationExecutor.java:31) at org.gradle.groovy.scripts.internal.BuildOperationBackedScriptCompilationHandler.compileToDir(BuildOperationBackedScriptCompilationHandler.java:51) at org.gradle.groovy.scripts.internal.FileCacheBackedScriptClassCompiler$CompileToCrossBuildCacheAction.execute(FileCacheBackedScriptClassCompiler.java:190) at org.gradle.groovy.scripts.internal.FileCacheBackedScriptClassCompiler$CompileToCrossBuildCacheAction.execute(FileCacheBackedScriptClassCompiler.java:170) at org.gradle.groovy.scripts.internal.FileCacheBackedScriptClassCompiler$ProgressReportingInitializer.execute(FileCacheBackedScriptClassCompiler.java:211) at org.gradle.groovy.scripts.internal.FileCacheBackedScriptClassCompiler$ProgressReportingInitializer.execute(FileCacheBackedScriptClassCompiler.java:194) at org.gradle.cache.internal.DefaultPersistentDirectoryCache$Initializer.initialize(DefaultPersistentDirectoryCache.java:100) at org.gradle.cache.internal.FixedSharedModeCrossProcessCacheAccess$1.run(FixedSharedModeCrossProcessCacheAccess.java:86) at org.gradle.cache.internal.DefaultFileLockManager$DefaultFileLock.doWriteAction(DefaultFileLockManager.java:215) at org.gradle.cache.internal.DefaultFileLockManager$DefaultFileLock.writeFile(DefaultFileLockManager.java:205) at org.gradle.cache.internal.FixedSharedModeCrossProcessCacheAccess.open(FixedSharedModeCrossProcessCacheAccess.java:83) at org.gradle.cache.internal.DefaultCacheAccess.open(DefaultCacheAccess.java:139) at org.gradle.cache.internal.DefaultPersistentDirectoryStore.open(DefaultPersistentDirectoryStore.java:78) at org.gradle.cache.internal.DefaultPersistentDirectoryStore.open(DefaultPersistentDirectoryStore.java:42) at org.gradle.cache.internal.DefaultCacheFactory.doOpen(DefaultCacheFactory.java:95) at org.gradle.cache.internal.DefaultCacheFactory.open(DefaultCacheFactory.java:68) at org.gradle.cache.internal.DefaultCacheRepository$PersistentCacheBuilder.open(DefaultCacheRepository.java:126) at org.gradle.groovy.scripts.internal.FileCacheBackedScriptClassCompiler.compile(FileCacheBackedScriptClassCompiler.java:116) at org.gradle.groovy.scripts.internal.CrossBuildInMemoryCachingScriptClassCache.getOrCompile(CrossBuildInMemoryCachingScriptClassCache.java:50) at org.gradle.groovy.scripts.internal.BuildScopeInMemoryCachingScriptClassCompiler.compile(BuildScopeInMemoryCachingScriptClassCompiler.java:50) at org.gradle.groovy.scripts.DefaultScriptCompilerFactory$ScriptCompilerImpl.compile(DefaultScriptCompilerFactory.java:49) at org.gradle.configuration.DefaultScriptPluginFactory$ScriptPluginImpl.apply(DefaultScriptPluginFactory.java:110) at org.gradle.configuration.BuildOperationScriptPlugin$1.run(BuildOperationScriptPlugin.java:65) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:395) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:387) at org.gradle.internal.operations.DefaultBuildOperationExecutor$1.execute(DefaultBuildOperationExecutor.java:157) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:242) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:150) at org.gradle.internal.operations.DefaultBuildOperationExecutor.run(DefaultBuildOperationExecutor.java:84) at org.gradle.internal.operations.DelegatingBuildOperationExecutor.run(DelegatingBuildOperationExecutor.java:31) at org.gradle.configuration.BuildOperationScriptPlugin.lambda$apply$0(BuildOperationScriptPlugin.java:62) at org.gradle.configuration.internal.DefaultUserCodeApplicationContext.apply(DefaultUserCodeApplicationContext.java:43) at org.gradle.configuration.BuildOperationScriptPlugin.apply(BuildOperationScriptPlugin.java:62) at org.gradle.configuration.project.BuildScriptProcessor$1.run(BuildScriptProcessor.java:45) at org.gradle.internal.Factories$1.create(Factories.java:26) at org.gradle.api.internal.project.DefaultProjectStateRegistry$ProjectStateImpl.withMutableState(DefaultProjectStateRegistry.java:245) at org.gradle.api.internal.project.DefaultProjectStateRegistry$ProjectStateImpl.withMutableState(DefaultProjectStateRegistry.java:226) at org.gradle.configuration.project.BuildScriptProcessor.execute(BuildScriptProcessor.java:42) at org.gradle.configuration.project.BuildScriptProcessor.execute(BuildScriptProcessor.java:26) at org.gradle.configuration.project.ConfigureActionsProjectEvaluator.evaluate(ConfigureActionsProjectEvaluator.java:35) at org.gradle.configuration.project.LifecycleProjectEvaluator$EvaluateProject$1.run(LifecycleProjectEvaluator.java:102) at org.gradle.internal.Factories$1.create(Factories.java:26) at org.gradle.internal.work.DefaultWorkerLeaseService.withLocks(DefaultWorkerLeaseService.java:180) at org.gradle.internal.work.StopShieldingWorkerLeaseService.withLocks(StopShieldingWorkerLeaseService.java:40) at org.gradle.api.internal.project.DefaultProjectStateRegistry$ProjectStateImpl.withProjectLock(DefaultProjectStateRegistry.java:271) at org.gradle.api.internal.project.DefaultProjectStateRegistry$ProjectStateImpl.withMutableState(DefaultProjectStateRegistry.java:265) at org.gradle.api.internal.project.DefaultProjectStateRegistry$ProjectStateImpl.withMutableState(DefaultProjectStateRegistry.java:226) at org.gradle.configuration.project.LifecycleProjectEvaluator$EvaluateProject.run(LifecycleProjectEvaluator.java:91) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:395) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:387) at org.gradle.internal.operations.DefaultBuildOperationExecutor$1.execute(DefaultBuildOperationExecutor.java:157) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:242) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:150) at org.gradle.internal.operations.DefaultBuildOperationExecutor.run(DefaultBuildOperationExecutor.java:84) at org.gradle.internal.operations.DelegatingBuildOperationExecutor.run(DelegatingBuildOperationExecutor.java:31) at org.gradle.configuration.project.LifecycleProjectEvaluator.evaluate(LifecycleProjectEvaluator.java:63) at org.gradle.api.internal.project.DefaultProject.evaluate(DefaultProject.java:710) at org.gradle.api.internal.project.DefaultProject.evaluate(DefaultProject.java:145) at org.gradle.execution.TaskPathProjectEvaluator.configure(TaskPathProjectEvaluator.java:36) at org.gradle.execution.TaskPathProjectEvaluator.configureHierarchy(TaskPathProjectEvaluator.java:62) at org.gradle.configuration.DefaultProjectsPreparer.prepareProjects(DefaultProjectsPreparer.java:61) at org.gradle.configuration.BuildOperatingFiringProjectsPreparer$ConfigureBuild.run(BuildOperatingFiringProjectsPreparer.java:52) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:395) at org.gradle.internal.operations.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:387) at org.gradle.internal.operations.DefaultBuildOperationExecutor$1.execute(DefaultBuildOperationExecutor.java:157) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:242) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:150) at org.gradle.internal.operations.DefaultBuildOperationExecutor.run(DefaultBuildOperationExecutor.java:84) at org.gradle.internal.operations.DelegatingBuildOperationExecutor.run(DelegatingBuildOperationExecutor.java:31) at org.gradle.configuration.BuildOperatingFiringProjectsPreparer.prepareProjects(BuildOperatingFiringProjectsPreparer.java:40) at org.gradle.initialization.DefaultGradleLauncher.prepareProjects(DefaultGradleLauncher.java:235) at org.gradle.initialization.DefaultGradleLauncher.doClassicBuildStages(DefaultGradleLauncher.java:170) at org.gradle.initialization.DefaultGradleLauncher.doBuildStages(DefaultGradleLauncher.java:155) at org.gradle.initialization.DefaultGradleLauncher.executeTasks(DefaultGradleLauncher.java:131) at org.gradle.internal.invocation.GradleBuildController$1.create(GradleBuildController.java:72) at org.gradle.internal.invocation.GradleBuildController$1.create(GradleBuildController.java:67) at org.gradle.internal.work.DefaultWorkerLeaseService.withLocks(DefaultWorkerLeaseService.java:180) at org.gradle.internal.work.StopShieldingWorkerLeaseService.withLocks(StopShieldingWorkerLeaseService.java:40) at org.gradle.internal.invocation.GradleBuildController.doBuild(GradleBuildController.java:67) at org.gradle.internal.invocation.GradleBuildController.run(GradleBuildController.java:56) at org.gradle.tooling.internal.provider.ExecuteBuildActionRunner.run(ExecuteBuildActionRunner.java:31) at org.gradle.launcher.exec.ChainingBuildActionRunner.run(ChainingBuildActionRunner.java:35) at org.gradle.launcher.exec.BuildOutcomeReportingBuildActionRunner.run(BuildOutcomeReportingBuildActionRunner.java:63) at org.gradle.tooling.internal.provider.ValidatingBuildActionRunner.run(ValidatingBuildActionRunner.java:32) at org.gradle.tooling.internal.provider.FileSystemWatchingBuildActionRunner.run(FileSystemWatchingBuildActionRunner.java:52) at org.gradle.launcher.exec.BuildCompletionNotifyingBuildActionRunner.run(BuildCompletionNotifyingBuildActionRunner.java:41) at org.gradle.launcher.exec.RunAsBuildOperationBuildActionRunner$3.call(RunAsBuildOperationBuildActionRunner.java:49) at org.gradle.launcher.exec.RunAsBuildOperationBuildActionRunner$3.call(RunAsBuildOperationBuildActionRunner.java:44) at org.gradle.internal.operations.DefaultBuildOperationExecutor$CallableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:409) at org.gradle.internal.operations.DefaultBuildOperationExecutor$CallableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:399) at org.gradle.internal.operations.DefaultBuildOperationExecutor$1.execute(DefaultBuildOperationExecutor.java:157) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:242) at org.gradle.internal.operations.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:150) at org.gradle.internal.operations.DefaultBuildOperationExecutor.call(DefaultBuildOperationExecutor.java:94) at org.gradle.internal.operations.DelegatingBuildOperationExecutor.call(DelegatingBuildOperationExecutor.java:36) at org.gradle.launcher.exec.RunAsBuildOperationBuildActionRunner.run(RunAsBuildOperationBuildActionRunner.java:44) at org.gradle.launcher.exec.InProcessBuildActionExecuter$1.transform(InProcessBuildActionExecuter.java:50) at org.gradle.launcher.exec.InProcessBuildActionExecuter$1.transform(InProcessBuildActionExecuter.java:47) at org.gradle.composite.internal.DefaultRootBuildState.run(DefaultRootBuildState.java:87) at org.gradle.launcher.exec.InProcessBuildActionExecuter.execute(InProcessBuildActionExecuter.java:47) at org.gradle.launcher.exec.InProcessBuildActionExecuter.execute(InProcessBuildActionExecuter.java:31) at org.gradle.launcher.exec.BuildTreeScopeBuildActionExecuter.execute(BuildTreeScopeBuildActionExecuter.java:44) at org.gradle.launcher.exec.BuildTreeScopeBuildActionExecuter.execute(BuildTreeScopeBuildActionExecuter.java:29) at org.gradle.tooling.internal.provider.ContinuousBuildActionExecuter.execute(ContinuousBuildActionExecuter.java:87) at org.gradle.tooling.internal.provider.ContinuousBuildActionExecuter.execute(ContinuousBuildActionExecuter.java:55) at org.gradle.tooling.internal.provider.SubscribableBuildActionExecuter.execute(SubscribableBuildActionExecuter.java:60) at org.gradle.tooling.internal.provider.SubscribableBuildActionExecuter.execute(SubscribableBuildActionExecuter.java:38) at org.gradle.tooling.internal.provider.SessionScopeBuildActionExecuter.execute(SessionScopeBuildActionExecuter.java:68) at org.gradle.tooling.internal.provider.SessionScopeBuildActionExecuter.execute(SessionScopeBuildActionExecuter.java:38) at org.gradle.tooling.internal.provider.GradleThreadBuildActionExecuter.execute(GradleThreadBuildActionExecuter.java:37) at org.gradle.tooling.internal.provider.GradleThreadBuildActionExecuter.execute(GradleThreadBuildActionExecuter.java:26) at org.gradle.tooling.internal.provider.StartParamsValidatingActionExecuter.execute(StartParamsValidatingActionExecuter.java:60) at org.gradle.tooling.internal.provider.StartParamsValidatingActionExecuter.execute(StartParamsValidatingActionExecuter.java:32) at org.gradle.tooling.internal.provider.SessionFailureReportingActionExecuter.execute(SessionFailureReportingActionExecuter.java:56) at org.gradle.tooling.internal.provider.SessionFailureReportingActionExecuter.execute(SessionFailureReportingActionExecuter.java:42) at org.gradle.tooling.internal.provider.SetupLoggingActionExecuter.execute(SetupLoggingActionExecuter.java:48) at org.gradle.tooling.internal.provider.SetupLoggingActionExecuter.execute(SetupLoggingActionExecuter.java:32) at org.gradle.launcher.daemon.server.exec.ExecuteBuild.doBuild(ExecuteBuild.java:68) at org.gradle.launcher.daemon.server.exec.BuildCommandOnly.execute(BuildCommandOnly.java:37) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.WatchForDisconnection.execute(WatchForDisconnection.java:39) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.ResetDeprecationLogger.execute(ResetDeprecationLogger.java:29) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.RequestStopIfSingleUsedDaemon.execute(RequestStopIfSingleUsedDaemon.java:35) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.ForwardClientInput$2.create(ForwardClientInput.java:78) at org.gradle.launcher.daemon.server.exec.ForwardClientInput$2.create(ForwardClientInput.java:75) at org.gradle.util.Swapper.swap(Swapper.java:38) at org.gradle.launcher.daemon.server.exec.ForwardClientInput.execute(ForwardClientInput.java:75) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.LogAndCheckHealth.execute(LogAndCheckHealth.java:55) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.LogToClient.doBuild(LogToClient.java:63) at org.gradle.launcher.daemon.server.exec.BuildCommandOnly.execute(BuildCommandOnly.java:37) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.EstablishBuildEnvironment.doBuild(EstablishBuildEnvironment.java:82) at org.gradle.launcher.daemon.server.exec.BuildCommandOnly.execute(BuildCommandOnly.java:37) at org.gradle.launcher.daemon.server.api.DaemonCommandExecution.proceed(DaemonCommandExecution.java:104) at org.gradle.launcher.daemon.server.exec.StartBuildOrRespondWithBusy$1.run(StartBuildOrRespondWithBusy.java:52) at org.gradle.launcher.daemon.server.DaemonStateCoordinator$1.run(DaemonStateCoordinator.java:297) at org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:64) at org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:48) at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144) at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:642) at org.gradle.internal.concurrent.ThreadFactoryImpl$ManagedThreadRunnable.run(ThreadFactoryImpl.java:56) at java.base/java.lang.Thread.run(Thread.java:1575) 1 error

Помилка Unsupported class file major version 67 вказує, що Gradle намагається компілювати проєкт із JDK, який є надто новим (версія 23, оскільки major version 67 відповідає Java 23). Твій build.gradle налаштовано на sourceCompatibility = '11', тому потрібна Java 11.

Рішення

* Перевір JDK: Виконай:

bash

java -version

Якщо показує Java 23 (або іншу версію, крім 11), встанови JDK 11.

* Встанови JDK 11:
  + Завантаж OpenJDK 11 (наприклад, від Adoptium: <https://adoptium.net/>).
  + Встанови та налаштуй змінну середовища JAVA\_HOME:

bash

export JAVA\_HOME=/path/to/jdk-11

(На Windows: налаштуй у системних змінних).

* Онови Gradle для сумісності: У gradle/wrapper/gradle-wrapper.properties вкажи Gradle 7.6 (сумісний із Java 11):

properties

distributionUrl=https\://services.gradle.org/distributions/gradle-7.6-bin.zip

Онови Gradle:

bash

./gradlew wrapper

* Очисти та скомпілюй:

bash

./gradlew clean build

Примітка

* Переконайся, що шлях до JDK 11 вказаний у PATH і JAVA\_HOME.
* Якщо використовуєш IDE, налаштуй проєкт на JDK 11 у налаштуваннях (наприклад, у IntelliJ: File -> Project Structure -> SDK).

Якщо помилка залишиться, напиши, яку версію Java показує java -version.

Я запустив команду wrapper Бачу такий результат: Welcome to Gradle 7.6!Here are the highlights of this release:

* Added support for Java 19.
* Introduced --rerun flag for individual task rerun.
* Improved dependency block for test suites to be strongly typed.
* Added a pluggable system for Java toolchains provisioning.

For more details see https://docs.gradle.org/7.6/release-notes.htmlStarting a Gradle Daemon (subsequent builds will be faster)FAILURE: Build failed with an exception.

* What went wrong: Could not open settings generic class cache for settings file 'C:\repository\telegram-bot-standard\settings.gradle' (C:\Users\belda\.gradle\caches\7.6\scripts\3kg9v6lquu5xp6eh13ixuzm2y).> BUG! exception in phase 'semantic analysis' in source unit '\_BuildScript\_' Unsupported class file major version 67

Ось мій файл settings.gradle, він має лише один рядок: rootProject.name = 'telegram-standard-bot'Що потрібно змінити у ньому?

Помилка Unsupported class file major version 67 знову вказує на те, що Gradle використовує JDK 23, який несумісний із твоїм проєктом, налаштованим на Java 11. Проблема не в самому файлі settings.gradle — він коректний із рядком:

gradle

rootProject.name = 'telegram-standard-bot'

Проблема в тому, що Gradle запускається з JDK 23, а не JDK 11. Треба змусити Gradle використовувати правильну версію Java.

Рішення

* Перевір версію Java: Виконай:

bash

java -version

Якщо показує Java 23, потрібно змінити JDK.

* Встанови JDK 11:
  + Завантаж OpenJDK 11 (наприклад, із <https://adoptium.net/>).
  + Встанови та налаштуй змінну середовища JAVA\_HOME:

bash

set JAVA\_HOME=C:\path\to\jdk-11

(На Windows: додай у системні змінні через Панель керування).

* Налаштуй Gradle для використання JDK 11: У файлі gradle.properties (створи, якщо немає, у корені проєкту) додай:

properties

org.gradle.java.home=C:/path/to/jdk-11

Заміни C:/path/to/jdk-11 на шлях до твого JDK 11.

* Очисти кеш Gradle: Видали кеш, щоб уникнути старих даних:

bash

rmdir /s /q %USERPROFILE%\.gradle\caches

* Перезапусти Gradle: Виконай:

bash

gradlew clean build

Чому settings.gradle не винен?

Файл settings.gradle лише задає ім’я проєкту і не впливає на компіляцію. Помилка виникає через несумісність версії JDK, яку Gradle використовує для парсингу скриптів.

Якщо не спрацює

* Перевір, чи JAVA\_HOME і PATH вказують на JDK 11:

bash

echo %JAVA\_HOME%

java -version

* Якщо JDK 11 не встановлено, завантаж його та онови JAVA\_HOME.
* Напиши, яку версію показує java -version, якщо проблема залишиться.

Нічого в settings.gradle міняти не треба.

дізнатися про Java 17

використання Maven замість Gradle