console.clear();

console.log("Starting...");

require("./setting/config");

const {

default: makeWASocket,

prepareWAMessageMedia,

useMultiFileAuthState,

DisconnectReason,

fetchLatestBaileysVersion,

makeInMemoryStore,

generateWAMessageFromContent,

generateWAMessageContent,

jidDecode,

proto,

relayWAMessage,

getContentType,

getAggregateVotesInPollMessage,

downloadContentFromMessage,

fetchLatestWaWebVersion,

InteractiveMessage,

makeCacheableSignalKeyStore,

Browsers,

generateForwardMessageContent,

MessageRetryMap

} = require("@whiskeysockets/baileys");

const pino = require("pino");

const readline = require("readline");

const fs = require("fs");

const {

Boom

} = require("@hapi/boom");

const {

color

} = require("./lib/color");

const {

smsg,

sendGmail,

formatSize,

isUrl,

generateMessageTag,

getBuffer,

getSizeMedia,

runtime,

fetchJson,

sleep

} = require("./lib/myfunction");

const usePairingCode = true;

const question = \_0x388dc2 => {

const \_0x5a14de = readline.createInterface({

input: process.stdin,

output: process.stdout

});

return new Promise(\_0x3e26ae => {

\_0x5a14de.question(\_0x388dc2, \_0x3e26ae);

});

};

const store = makeInMemoryStore({

logger: pino().child({

level: "silent",

stream: "store"

})

});

async function korpsstart() {

const {

state: \_0x1a082c,

saveCreds: \_0x36dab8

} = await useMultiFileAuthState("session");

const \_0x437fc5 = makeWASocket({

printQRInTerminal: !usePairingCode,

syncFullHistory: true,

markOnlineOnConnect: true,

connectTimeoutMs: 60000,

defaultQueryTimeoutMs: 0,

keepAliveIntervalMs: 10000,

generateHighQualityLinkPreview: true,

patchMessageBeforeSending: \_0x542e7c => {

const \_0x588119 = !!\_0x542e7c.buttonsMessage || !!\_0x542e7c.templateMessage || !!\_0x542e7c.listMessage;

if (\_0x588119) {

\_0x542e7c = {

viewOnceMessage: {

message: {

messageContextInfo: {

deviceListMetadataVersion: 2,

deviceListMetadata: {}

},

...\_0x542e7c

}

}

};

}

return \_0x542e7c;

},

version: (await (await fetch("https://raw.githubusercontent.com/WhiskeySockets/Baileys/master/src/Defaults/baileys-version.json")).json()).version,

browser: ["Ubuntu", "Chrome", "20.0.04"],

logger: pino({

level: "fatal"

}),

auth: {

creds: \_0x1a082c.creds,

keys: makeCacheableSignalKeyStore(\_0x1a082c.keys, pino().child({

level: "silent",

stream: "store"

}))

}

});

if (usePairingCode && !\_0x437fc5.authState.creds.registered) {

const \_0x22c0a5 = await question("╭━〔 🕊️ **SHADOW** 𝐁𝐮𝐠 〕━⬣\n✮ 𝙋𝘼𝙄𝙍𝙄𝙉𝙂 𝘾𝙊𝘿𝙀\n✮ 𝙀𝙉𝙏𝙀𝙍 **YOUR** 𝙉𝙐𝙈𝘽𝙀𝙍 \n╭━━━━━━━━━━━━━❦\n┇**SHADOW** 𝗕𝗨𝗚 𝗕𝗢𝗧 : 50933xxxxxx\n╰━━━━━━━━━━━━⬣");

const \_0x21cafd = await \_0x437fc5.requestPairingCode(\_0x22c0a5.trim());

console.log("Code : " + \_0x21cafd);

}

store.bind(\_0x437fc5.ev);

\_0x437fc5.ev.on("messages.upsert", async (\_0x44353b, \_0x559dc3) => {

try {

const \_0x38374a = \_0x44353b.messages[0];

if (!\_0x38374a.message) {

return;

}

\_0x38374a.message = Object.keys(\_0x38374a.message)[0] === "ephemeralMessage" ? \_0x38374a.message.ephemeralMessage.message : \_0x38374a.message;

if (\_0x38374a.key && \_0x38374a.key.remoteJid === "status@broadcast") {

return;

}

if (!\_0x437fc5.public && !\_0x38374a.key.fromMe && \_0x44353b.type === "notify") {

return;

}

if (\_0x38374a.key.id.startsWith("BAE5") && \_0x38374a.key.id.length === 16) {

return;

}

if (\_0x38374a.key.id.startsWith("FatihArridho\_")) {

return;

}

const \_0x47a0a6 = smsg(\_0x437fc5, \_0x38374a, store);

require("./Storm.js")(\_0x437fc5, \_0x47a0a6, \_0x44353b, store);

} catch (\_0x125438) {

console.log(\_0x125438);

}

});

\_0x437fc5.decodeJid = \_0x2d7e44 => {

if (!\_0x2d7e44) {

return \_0x2d7e44;

}

if (/:\d+@/gi.test(\_0x2d7e44)) {

let \_0x35e136 = jidDecode(\_0x2d7e44) || {};

return \_0x35e136.user && \_0x35e136.server && \_0x35e136.user + "@" + \_0x35e136.server || \_0x2d7e44;

} else {

return \_0x2d7e44;

}

};

\_0x437fc5.ev.on("contacts.update", \_0x2765a8 => {

for (let \_0x52c8d4 of \_0x2765a8) {

let \_0x396179 = \_0x437fc5.decodeJid(\_0x52c8d4.id);

if (store && store.contacts) {

store.contacts[\_0x396179] = {

id: \_0x396179,

name: \_0x52c8d4.notify

};

}

}

});

global.idch1 = "120363419474272514@newsletter";

global.idch2 = "120363404759959596@newsletter";

\_0x437fc5.public = global.status;

\_0x437fc5.ev.on("connection.update", async \_0xefd6dc => {

const {

connection: \_0x2f31f4,

lastDisconnect: \_0x39d321

} = \_0xefd6dc;

if (\_0x2f31f4 === "close") {

const \_0xa10d18 = new Boom(\_0x39d321?.error)?.output.statusCode;

console.log(color(\_0x39d321.error, "deep pink"));

if (\_0x39d321.error == "") {

process.exit();

} else if (\_0xa10d18 === DisconnectReason.badSession) {

console.log(color("Bad Session File, Please Delete Session and Scan Again"));

process.exit();

} else if (\_0xa10d18 === DisconnectReason.connectionClosed) {

console.log(color("[SYSTEM]", "white"), color("Connection closed, reconnecting...", "deeppink"));

process.exit();

} else if (\_0xa10d18 === DisconnectReason.connectionLost) {

console.log(color("[SYSTEM]", "white"), color("Connection lost, trying to reconnect", "deeppink"));

process.exit();

} else if (\_0xa10d18 === DisconnectReason.connectionReplaced) {

console.log(color("Connection Replaced, Another New Session Opened, Please Close Current Session First"));

\_0x437fc5.logout();

} else if (\_0xa10d18 === DisconnectReason.loggedOut) {

console.log(color("Device Logged Out, Please Scan Again And Run."));

\_0x437fc5.logout();

} else if (\_0xa10d18 === DisconnectReason.restartRequired) {

console.log(color("Restart Required, Restarting..."));

await korpsstart();

} else if (\_0xa10d18 === DisconnectReason.timedOut) {

console.log(color("Connection TimedOut, Reconnecting..."));

korpsstart();

}

} else if (\_0x2f31f4 === "connecting") {

console.log(color("Connect . . . "));

} else if (\_0x2f31f4 === "open") {

\_0x437fc5.newsletterFollow(global.idch1);

\_0x437fc5.newsletterFollow(global.idch2);

console.log(color("Bot Connected Successfully"));

}

});

\_0x437fc5.sendText = (\_0x47df39, \_0x526e9f, \_0x29776f = "", \_0x798f4b) => \_0x437fc5.sendMessage(\_0x47df39, {

text: \_0x526e9f,

...\_0x798f4b

}, {

quoted: \_0x29776f

});

\_0x437fc5.downloadMediaMessage = async \_0x4d3744 => {

let \_0x344120 = (\_0x4d3744.msg || \_0x4d3744).mimetype || "";

let \_0x4a63d6 = \_0x4d3744.mtype ? \_0x4d3744.mtype.replace(/Message/gi, "") : \_0x344120.split("/")[0];

const \_0x25dd16 = await downloadContentFromMessage(\_0x4d3744, \_0x4a63d6);

let \_0x1e0125 = Buffer.from([]);

for await (const \_0x44536e of \_0x25dd16) {

\_0x1e0125 = Buffer.concat([\_0x1e0125, \_0x44536e]);

}

return \_0x1e0125;

};

\_0x437fc5.ev.on("creds.update", \_0x36dab8);

return \_0x437fc5;

}

korpsstart();

let file = require.resolve(\_\_filename);

require("fs").watchFile(file, () => {

require("fs").unwatchFile(file);

console.log("[0;32m" + \_\_filename + " [1;32mupdated![0m");

delete require.cache[file];

require(file);

});