```
/*
Don't change credits🥲🚶
Sourcecode https://github.com/pepesir/Bosco

Want to recode? yes, it's up to you as long as you don't change the watermark
Don't sell the SC 😐👌
*/
const {
    WAConnection,
        MessageType,
        Presence,
        MessageOptions,
        Mimetype,
        WALocationMessage,
        WA_MESSAGE_STUB_TYPES,
        WA_DEFAULT_EPHEMERAL,
        ReconnectMode,
        ProxyAgent,
        ChatModification,
        GroupSettingChange,
        waChatKey,
        mentionedJid,
        processTime,
        Browsers,
} = require("@adiwajshing/baileys")
const moment = require("moment-timezone")
const speed = require('performance-now')
const { spawn, exec, execSync } = require("child_process")
const ffmpeg = require('fluent-ffmpeg')
const twitterGetUrl = require("twitter-url-direct")
const brainly = require('brainly-scraper')
const translate = require("@vitalets/google-translate-api");
const { EmojiAPI } = require("emoji-api")
const emoji = new EmojiAPI()
const Math_js = require('mathjs')
const _gis = require('g-i-s')
const fetch = require('node-fetch');
const request = require('request');
const yts = require( 'yt-search')
const ms = require('parse-ms')
const toMs = require('ms')
const axios = require("axios")
const fs = require("fs-extra")
const crypto = require('crypto')
const { promisify, util } = require('util')
const qrcodes = require('qrcode');
const googleIt = require('google-it')
const os = require('os');
const hx = require('hxz-api')

const { Menu, rulesBot } = require('./message/help.js')
const { getBuffer, getGroupAdmins, generateMessageID, getRandom, runtime, pickRandom, sleep } =
require('./lib/myfunc')
const { fetchJson, getBase64, kyun, createExif } = require('./lib/fetch')
const { color, bgcolor } = require('./lib/color')
const { mess } = require('./message/mess')
const { Toxic } = require('./lib/Toxic.js')
const { cmdadd } = require('./lib/totalcmd.js')
const { uptotele, uploadFile, RESTfulAPI, uploadImages } = require('./lib/uploadimage')
const { lirikLagu} = require('./lib/lirik.js')
const { mediafireDl } = require('./lib/mediafire.js')
const { webp2gifFile, igDownloader, TiktokDownloader } = require("./lib/gif.js")
const { y2mateA, y2mateV } = require('./lib/y2mate')
const { webp2mp4File } = require('./lib/webp2mp4')
const { jadibot, stopjadibot, listjadibot } = require('./lib/jadibot')
const truth = JSON.parse(fs.readFileSync('./database/truth.json'))
const dare = JSON.parse(fs.readFileSync('./database/dare.json'))
const atm = require("./lib/atm");
const level = require("./lib/level");
const afk = JSON.parse(fs.readFileSync('./lib/off.json'))
const { isAfk, cekafk, addafk } = require('./lib/offline')
const a = '▢'

hit_today = []
banChats = false
offline = false

let fakeimage = fs.readFileSync("./media/wpmobile.jpg")
let setting = JSON.parse(fs.readFileSync('./setting.json'))


prefix = setting.prefix
```

```
owner = setting.owner
lolkey = setting.lolkey
ownerName = setting.ownerName
fake = setting.fake
autorespon = false
waktu = '-'
alasan = '-'

const Exif = require('./lib/exif')
const exif = new Exif()

// Database
let register = JSON.parse(fs.readFileSync('./database/register.json'))
let _scommand = JSON.parse(fs.readFileSync('./database/scommand.json'))
let config
let _leveling = JSON.parse(fs.readFileSync('./database/leveling.json'))
let _level = JSON.parse(fs.readFileSync('./database/level.json'))
let _uang = JSON.parse(fs.readFileSync('./database/uang.json'))
let antilink = JSON.parse(fs.readFileSync('./database/antilink.json'));
const mute = JSON.parse(fs.readFileSync('./database/mute.json'))
const setik = JSON.parse(fs.readFileSync('./database/setik.json'))
const  vien = JSON.parse(fs.readFileSync('./database/vien.json'))
const  imagi = JSON.parse(fs.readFileSync('./database/imagi.json'))
const  videonye = JSON.parse(fs.readFileSync('./database/video.json'))
const autosticker = JSON.parse(fs.readFileSync('./database/autosticker.json'))
const kickarea = JSON.parse(fs.readFileSync('./database/kickarea.json'))

// Sticker Cmd
const addCmd = (id, command) => {
    const obj = { id: id, chats: command }
    _scommand.push(obj)
    fs.writeFileSync('./database/scommand.json', JSON.stringify(_scommand))
}

const getCommandPosition = (id) => {
    let position = null
    Object.keys(_scommand).forEach((i) => {
        if (_scommand[i].id === id) {
            position = i
        }
    })
    if (position !== null) {
        return position
    }
}

const getCmd = (id) => {
    let position = null
    Object.keys(_scommand).forEach((i) => {
        if (_scommand[i].id === id) {
            position = i
        }
    })
    if (position !== null) {
        return _scommand[position].chats
    }
}


const checkSCommand = (id) => {
    let status = false
    Object.keys(_scommand).forEach((i) => {
        if (_scommand[i].id === id) {
            status = true
        }
    })
    return status
}
module.exports = m = async (m) => {
        try {
if (m.key.remoteJid == 'status@broadcast') return
if (!m.key.fromMe && m.key.fromMe) return
m.message = (Object.keys(m.message)[0] === 'ephemeralMessage') ? m.message.ephemeralMessage.message :
m.message
const jamu = moment.tz('Asia/Kolkata').format('HH:mm:ss')
let d = new Date
let locale = 'id'
const gmtu = new Date(0).getTime() - new Date('1 Januari 2021').getTime()
const wetonj = ['Pahing', 'Pon','Wage','Kliwon','Legi'][Math.floor(((d * 1) + gmt) / 84600000) % 5]
const weeku = d.toLocaleDateString(locale, { weekday: 'long' })
const calenderh = d.toLocaleDateString(locale, {
day: 'numeric',
month: 'long',
year: 'numeric'
})
let keynye = m.key
let c = bosco.chats.get(keynye.remoteJid)
let a = c.messages.dict[`${keynye.id}|${keynye.fromMe ? 1: 0}`]
let contennye = bosco.generateForwardMessageContent(a, false)
} catch {
}
}


module.exports = bosco = async (bosco, mek) => {
        try {
        if (!mek.hasNewMessage) return
        mek = mek.messages.all()[0]
                if (!mek.message) return
                if (mek.key && mek.key.remoteJid == 'status@broadcast') return
                global.blocked
                global.prefix
                mek.message = (Object.keys(mek.message)[0] === 'ephemeralMessage') ?
mek.message.ephemeralMessage.message : mek.message
                const { text, extendedText, contact, location, liveLocation, image, video, sticker,
document, audio, product } = MessageType
                const time = moment().tz('Asia/Kolkata').format('HH:mm:ss')
                const wib = moment.tz('Asia/Kolkata').format('HH:mm:ss')
                const content = JSON.stringify(mek.message)
                const from = mek.key.remoteJid
                const type = Object.keys(mek.message)[0]
                const antibot = m.isBaileys
        const cmd = (type === 'conversation' && mek.message.conversation) ? mek.message.conversation :
(type == 'imageMessage') && mek.message.imageMessage.caption ? mek.message.imageMessage.caption : (type ==
'videoMessage') && mek.message.videoMessage.caption ? mek.message.videoMessage.caption : (type ==
'extendedTextMessage') && mek.message.extendedTextMessage.text ? mek.message.extendedTextMessage.text :
''.slice(1).trim().split(/ +/).shift().toLowerCase()
        body = (type === 'conversation' && mek.message.conversation.startsWith(prefix)) ?
mek.message.conversation : (type == 'imageMessage') && mek.message[type].caption.startsWith(prefix) ?
mek.message[type].caption : (type == 'videoMessage') && mek.message[type].caption.startsWith(prefix) ?
mek.message[type].caption : (type == 'extendedTextMessage') && mek.message[type].text.startsWith(prefix) ?
mek.message[type].text : (type == 'listResponseMessage') &&
mek.message[type].singleSelectReply.selectedRowId ? mek.message[type].singleSelectReply.selectedRowId :
(type == 'buttonsResponseMessage') && mek.message[type].selectedButtonId ?
mek.message[type].selectedButtonId : (type == 'stickerMessage') &&
(getCmd(mek.message[type].fileSha256.toString('base64')) !== null &&
getCmd(mek.message[type].fileSha256.toString('base64')) !== undefined) ?
getCmd(mek.message[type].fileSha256.toString('base64')) : ""
                budy = (type === 'conversation') ? mek.message.conversation : (type ===
'extendedTextMessage') ? mek.message.extendedTextMessage.text : ''
                const command = body.replace(prefix, '').trim().split(/ +/).shift().toLowerCase()
                const args = body.trim().split(/ +/).slice(1)
                hit_today.push(command)
                const arg = body.substring(body.indexOf(' ') + 1)
                const ar = args.map((v) => v.toLowerCase())
                const argz = body.trim().split(/ +/).slice(1)
                const isCmd = body.startsWith(prefix)
                if (isCmd) cmdadd()
                const totalhit = JSON.parse(fs.readFileSync('./database/totalcmd.json'))[0].totalcmd
        const q = args.join(' ')
        const c = args.join(' ')

        const botNumber = bosco.user.jid
        const ownerNumber = setting.ownerNumber
                const ownerName = setting.ownerName
                const botName = setting.botName
                const isGroup = from.endsWith('@g.us')
                const sender = mek.key.fromMe ? bosco.user.jid : mek.key.remoteJid.endsWith('@g.us') ?
mek.participant : mek.key.remoteJid
                const totalchat = await bosco.chats.all()
                isStc = Object.keys(mek.message)[0] == "stickerMessage" ?
mek.message.stickerMessage.fileSha256.toString('hex') : ""
            isStc = `${isStc}`
        const isStcQ = isStc !== "" && content.includes("extendedTextMessage") ||
        isStc !== "" && content.includes("conversation")
            const isStcMedia = isStc !== "" && content.includes("quotedMessage") &&
!content.includes("extendedTextMessage") || isStc !== "" && content.includes("quotedMessage") &&
!content.includes("conversation")
            const isStcVideo = isStcMedia && content.includes("videoMessage")
            const isStcImage = isStcMedia && content.includes("imageMessage")
            const isStcSticker = isStcMedia && content.includes("stickerMessage")
        const isStcTeks = isStcMedia && content.includes("quotedMessage")
        const isStcDocs = isStcMedia && content.includes("documentMessage")
        const isStcContact = isStcMedia && content.includes("contactMessage")
        const isStcAudio = isStcMedia && content.includes("audioMessage")
        const isStcLoca = isStcMedia && content.includes("locationMessage")
        const isStcTag = isStcMedia && content.includes("mentionedJid")
        const isStcReply = isStcMedia && content.includes("Message")
        const isStcProd = isStcMedia && content.includes("productMessage")
                const groupMetadata = isGroup ? await bosco.groupMetadata(from) : ''
                const groupName = isGroup ? groupMetadata.subject : ''
                const groupId = isGroup ? groupMetadata.jid : ''
                const groupMembers = isGroup ? groupMetadata.participants : ''
                const groupDesc = isGroup ? groupMetadata.desc : ''
                const groupOwner = isGroup ? groupMetadata.owner : ''
                const groupAdmins = isGroup ? getGroupAdmins(groupMembers) : ''
                const isBotGroupAdmins = groupAdmins.includes(botNumber) || false
                const isGroupAdmins = groupAdmins.includes(sender) || false
        const conts = mek.key.fromMe ? bosco.user.jid : bosco.contacts[sender] || { notify:
jid.replace(/@.+/, '') }
        const pushname = mek.key.fromMe ? bosco.user.name : conts.notify || conts.vname || conts.name || '-
'
        const mentionByTag = type == "extendedTextMessage" && mek.message.extendedTextMessage.contextInfo
!= null ? mek.message.extendedTextMessage.contextInfo.mentionedJid : []
        const mentionByreply = type == "extendedTextMessage" && mek.message.extendedTextMessage.contextInfo
!= null ? mek.message.extendedTextMessage.contextInfo.participant || "" : ""
        const mention = typeof(mentionByTag) == 'string' ? [mentionByTag] : mentionByTag
        mention != undefined ? mention.push(mentionByreply) : []
        const mentionUser = mention != undefined ? mention.filter(n => n) : []
                const dfrply = fs.readFileSync('./ds.jpg')
                const atibot = m.isBaileys
                const isRegister = register.includes(sender)
        const isOwner = ownerNumber.includes(sender)
        const isKickArea = isGroup ? kickarea.includes(from) : false
        const isAntiLink = isGroup ? antilink.includes(from) : false
        const isLevelingOn = isGroup ? _leveling.includes(from) : false
        const isMuted = isGroup ? mute.includes(from) : false
        const isAuto = isGroup ? autosticker.includes(from) : false



        // here button function
        selectedButton = (type == 'buttonsResponseMessage') ?
mek.message.buttonsResponseMessage.selectedButtonId : ''

        responseButton = (type == 'listResponseMessage') ? mek.message.listResponseMessage.title : ''

        if (antibot === true) return
                const catl = (teks) => {
             res = bosco.prepareMessageFromContent(from,{ "orderMessage": { "itemCount": 400, "message":
teks, "footerText": "*Pepe Ser*", "thumbnail": dfrply, "surface": 'CATALOG' }}, {quoted:ftrol})
             bosco.relayWAMessage(res)
        }
       const catlo = (teks) => {
             res = bosco.prepareMessageFromContent(from,{ "orderMessage": { "itemCount": 70000, "message":
teks, "footerText": "Made With Pepe", thumbnail: fs.readFileSync('./ds.jpg'), "surface": 'CATALOG' }},
{quoted:ftroli})
             bosco.relayWAMessage(res)
        }
        const grupinv = (teks) => {
                grup = bosco.prepareMessageFromContent(from, { "groupInviteMessage": { "groupJid":
'6288213840883-1616169743@g.us', "inviteCode": 'https://chat.whatsapp.com/BzhyWkAEU0t8oVl3s8p94m',
"groupName": `Bosco Family`, "footerText": "*Pepe Ser*", "jpegThumbnail": dfrply, "caption": teks}},
{quoted:fvideo})
            bosco.relayWAMessage(grup)
        }
        try {
                pporang = await bosco.getProfilePicture(`${sender.split('@')[0]}@s.whatsapp.net`)
                      } catch {
                pporang = 'https://i0.wp.com/www.gambarunik.id/wp-content/uploads/2019/06/Top-Gambar-Foto-
Profil-Kosong-Lucu-Tergokil-.jpg'
                      }
                const denis = await getBuffer(pporang)

        const isUrl = (url) => {
            return url.match(new RegExp(/https?:\/\/(www\.)?[-a-zA-Z0-9@:%._+~#=]{1,256}\.[a-zA-Z0-9()]
{1,6}\b([-a-zA-Z0-9()@:%_+.~#?&/=]*)/, 'gi'))
        }
        function monospace(string) {
            return '```' + string + '```'
        }
        function jsonformat(string) {
            return JSON.stringify(string, null, 2)
        }
        function randomNomor(angka){
            return Math.floor(Math.random() * angka) + 1
        }
        const reply = (teks) => {
              bosco.sendMessage(from, teks, text, { thumbnail: denis, sendEphemeral: true, quoted: mek,
```

```
contextInfo: { forwardingScore: 508, isForwarded: false, "externalAdReply": { "title": `${' '}Sᴜʙsᴄʀɪʙᴇ Yᴛ :
PEPE SIR${''}${''}`, "body": `Gʀᴏᴜᴘ Assɪsᴛᴇɴᴛ Bᴏᴛ`, "previewType": 'PHOTO', "thumbnailUrl": `${''}`,
"thumbnail": denis, "sourceUrl": `${''}`}},})
        }
        const sendMess = (hehe, teks) => {
           bosco.sendMessage(hehe, teks, text)
        }
        const mentions = (teks, memberr, id) => {
           (id == null || id == undefined || id == false) ? bosco.sendMessage(from, {text: teks.trim(),
jpegThumbnail: fs.readFileSync('./media/wpmobile.jpg')}, extendedText, { sendEphemeral: true, contextInfo:
{ "mentionedJid": memberr } }) : bosco.sendMessage(from, {text: teks.trim(), jpegThumbnail:
fs.readFileSync('./media/wpmobile.jpg')}, extendedText, { sendEphemeral: true, quoted: mek, contextInfo: {
"mentionedJid": memberr } })
        }
        const sendText = (from, text) => {
           bosco.sendMessage(from, text, MessageType.text)
        }
        const textImg = (teks) => {
           return bosco.sendMessage(from, teks, text, {quoted: mek, thumbnail:
fs.readFileSync('./media/wpmobile.jpg')})
        }

        const fakestatus = (teks) => {
            return bosco.sendMessage(from, teks, text, {
                quoted: {
                    key: {
                        fromMe: false,
                        participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {})
                    },
                    message: {
                        "imageMessage": {
                            "url": "https://mmg.whatsapp.net/d/f/At0x7ZdIvuicfjlf9oWS6A3AR9XPh0P-
hZIVPLsI70nM.enc",
                            "mimetype": "image/jpeg",
                            "caption": fake,
                            "fileSha256": "+Ia+Dwib70Y1CWRMAP9QLJKjIJt54fKycOfB2OEZbTU=",
                            "fileLength": "28777",
                            "height": 1080,
                            "width": 1079,
                            "mediaKey": "vXmRR7ZUeDWjXy5iQk17TrowBzuwRya0errAFnXxbGc=",
                            "fileEncSha256": "sR9D2RS5JSifw49HeBADguI23fWDz1aZu4faWG/CyRY=",
                            "directPath": "/v/t62.7118-
24/21427642_840952686474581_572788076332761430_n.enc?oh=3f57c1ba2fcab95f2c0bb475d72720ba&oe=602F3D69",
                            "mediaKeyTimestamp": "1610993486",
                            "jpegThumbnail": fs.readFileSync('./ds.jpg'),
                            "scansSidecar": "1W0XhfaAcDwc7xh1R8lca6Qg/1bB4naFCSngM2LKO2NoP5RI7K+zLw=="
                        }
                    }
                }
            })
        }
        const fakethumb = (teks, yes) => {
            return bosco.sendMessage(from, teks, image,
{thumbnail:fs.readFileSync('./stik/fake.jpg'),quoted:mek,caption:yes})
        }
        const fakegroup = (teks) => {
            return bosco.sendMessage(from, teks, text, {
                quoted: {
                    key: {
                        fromMe: false,
                        participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "6289523258649-
1604595598@g.us" } : {})
                    },
                    message: {
                        "imageMessage": {
                            "url": "https://mmg.whatsapp.net/d/f/At0x7ZdIvuicfjlf9oWS6A3AR9XPh0P-
hZIVPLsI70nM.enc",
                            "mimetype": "image/jpeg",
                            "caption": fake,
                            "fileSha256": "+Ia+Dwib70Y1CWRMAP9QLJKjIJt54fKycOfB2OEZbTU=",
                            "fileLength": "28777",
                            "height": 1080,
                            "width": 1079,
                            "mediaKey": "vXmRR7ZUeDWjXy5iQk17TrowBzuwRya0errAFnXxbGc=",
                            "fileEncSha256": "sR9D2RS5JSifw49HeBADguI23fWDz1aZu4faWG/CyRY=",
                            "directPath": "/v/t62.7118-
24/21427642_840952686474581_572788076332761430_n.enc?oh=3f57c1ba2fcab95f2c0bb475d72720ba&oe=602F3D69",
                            "mediaKeyTimestamp": "1610993486",
                            "jpegThumbnail": fs.readFileSync('./ds.jpg'),
                            "scansSidecar": "1W0XhfaAcDwc7xh1R8lca6Qg/1bB4naFCSngM2LKO2NoP5RI7K+zLw=="
                        }
                    }
                }
            })
        }
      const fgclink = (teks) => {
            bosco.sendMessage(from, teks, text, {
                quoted: {
                        key: {
                               fromMe: true,
                                participant: "0@s.whatsapp.net",
                                  remoteJid: "0@s.whatsapp.net"
                       },
                      message: {
                               "groupInviteMessage": {
                                       "groupJid": "6288213840883-1616169743@g.us",
                                       "inviteCode": "mememteeeekkeke",
                                       "groupName": ".bot",
                           "caption": `CMD EXCLUDED : \n NEW FEUTERS ADDED \n MADE BY PEPE SIR`,
                           'jpegThumbnail': fs.readFileSync(`ds.jpg`)
                                }
                       }
             }
       })
    }
    // TEXT WITH THUMBNAIL
const ftex = {
         key: {
          fromMe: false,
              participant: `0@s.whatsapp.net`, ...(from ?
         { remoteJid: "6289643739077-1613049930@g.us" } : {})
                },
         message: {
                "extendedTextMessage": {
                 "text": `hello bro ${pushname}`,
                 "title": `${pushname}`,
                 'jpegThumbnail': dfrply
                        }
                          }
                     }
       const fakeitem = (teks) => {
           return bosco.sendMessage(from, teks, text, {
                quoted: {
        key:{
                fromMe:false,
        participant:`0@s.whatsapp.net`, ...(from ? {
remoteJid :"status@broadcast" }: {})
                    },message:{"orderMessage":
{"orderId":"174238614569481","thumbnail":fs.readFileSync(`ds.jpg`),"itemCount":2021,"status":"INQUIRY","sur
face":"CATALOG","message":`${fake}`,"token":"AR6xBKbXZn0Xwmu76Ksyd7rnxI+Rx87HfinVlW4lwXa6JA=="}}},
contextInfo: {"forwardingScore":999,"isForwarded":true},sendEphemeral: true})}
                    //WAKTU
                        var ase = new Date();
                        var jamss = ase.getHours();
                         switch(jamss){
                case 0: jamss = "Midnight"; break;
                case 1: jamss = "Midnight"; break;
                case 2: jamss = "Midnight"; break;
                case 3: jamss = "Midnight"; break;
                case 4: jamss = "Midnight"; break;
                case 5: jamss = "Dawn"; break;
                case 6: jamss = "Morning"; break;
                case 7: jamss = "Morning"; break;
                case 8: jamss = "Morning"; break;
                case 9: jamss = "Morning"; break;
                case 10: jamss = "Morning"; break;
                case 11: jamss = "Afternoon"; break;
                case 12: jamss = "Zuhur"; break;
                case 13: jamss = "Afternoon"; break;
                case 14: jamss = "Afternoon"; break;
                case 15: jamss = "Asr"; break;
                case 16: jamss = "Afternoon"; break;
                case 17: jamss = "Evening"; break;
                case 18: jamss = "Maghrib"; break;
                case 19: jamss = "Isha"; break;
                case 20: jamss = "Night"; break;
                case 21: jamss = "Night"; break;
                case 22: jamss = "Midnight"; break;
                case 23: jamss = "Midnight"; break;
            }
            var tampilUcapan = "" + jamss;
            const jmo = moment.tz('Asia/Kolkata').format('DD/MM/YYYY')
            const jmn = moment.tz('Asia/Kolkata').format('hh:mm')
                                let d = new Date
                                let locale = 'en'
                                let gmt = new Date(0).getTime() - new Date('1 Januari 2021').getTime()
```

```
                                const weton = ['Pahing', 'Pon','Wage','Kliwon','Legi'][Math.floor(((d * 1)
+ gmt) / 84600000) % 5]
                                const week = d.toLocaleDateString(locale, { weekday: 'long' })
                                const calender = d.toLocaleDateString(locale, {
                                day: 'numeric',
                                month: 'long',
                                year: 'numeric'
                        })
                ///Button Location
/*const sendButLocation = async (id, text1, desc1, gam1, but = [], options = {}) => {
kma = gam1
mhan = await bosco.prepareMessage(from, kma, location)
/*const buttonMessages = {
locationMessage: mhan.message.locationMessage,
contentText: text1,
footerText: desc1,
buttons: but,
headerType: 6
/}
bosco.sendMessage(id, buttonMessages, MessageType.buttonsMessage, options)
/}*/
//Button ocument
const Sendbutdocument = async(id, text1, desc1, file1, doc1, but = [], options = {}) => {
media = file1
kma = doc1
mhan = await bosco.prepareMessage(from, media, document, kma)
const buttonMessages = {
documentMessage: mhan.message.documentMessage,
contentText: text1,
footerText: desc1,
buttons: but,
headerType: "DOCUMENT"
}
bosco.sendMessage(id, buttonMessages, MessageType.buttonsMessage, options)
}

// VIDEO
const fvid = {
         key: {
          fromMe: false,
              participant: `0@s.whatsapp.net`, ...(from ?
         { remoteJid: "6289643739077-1613049930@g.us" } : {})
                },
         message: {
                 "videoMessage": {
                 "title": `pepe sir`,
                 "h": `${tampilUcapan} ${pushname}`,
                 'duration': '99999',
                 'caption': `${tampilUcapan} ${pushname}`,
                 'jpegThumbnail': dfrply
                        }
                       }
                          }

        //FAKEREPLY PRODUCT
            const ftoko = {
                  key: {fromMe: false,participant: `0@s.whatsapp.net`, ...(from ? { remoteJid:
"16505434800@s.whatsapp.net" } : {})},message: {"productMessage": {"product": {"productImage":{"mimetype":
"image/jpeg","jpegThumbnail": fs.readFileSync(`./ds.jpg`)},"title": `▢ ${fake} ▢`,"description": "hehe",
"currencyCode": "US$","priceAmount1000": "9999999999","retailerId": "X - Dev Team","productImageCount":
1},"businessOwnerJid": `0@s.whatsapp.net`}}}
            //FAKE KONTAK
            const fkontak = {
                  key: {fromMe: false,participant: `0@s.whatsapp.net`, ...(from ? { remoteJid:
`0@s.whatsapp.net` } : {}) }, message: { 'contactMessage': { 'displayName': `${fake}`, 'vcard':
`BEGIN:VCARD\nVERSION:3.0\nN:XL;${pushname},;;;\nFN:${pushname},\nitem1.TEL;waid=${sender.split('@')
[0]}:${sender.split('@')[0]}\nitem1.X-ABLabel:Its Me Pepe Ser\nEND:VCARD`, 'jpegThumbnail':
fs.readFileSync('./ds.jpg')}}}
            //FAKE STICKER
            const fsticker = {
                  key: {fromMe: false,participant: `0@s.whatsapp.net`, ...(from ? { remoteJid:
"16505434800@s.whatsapp.net" } : {})},"message": {"stickerMessage": { "url":
"https://mmg.whatsapp.net/d/f/Am6FBfNf-E2f1VoGBXkPaNAy7L6Tw_HMavKrHEt48QM4.enc","fileSha256":
"Yfj8SW7liSEnDakvyVlXVZQ1LJBC9idn09X7KHe8HTc=","fileEncSha256":
"F854aUrzgAkBTOVULpne4oSIi6S04Jo56pjZEo+p+9U=","mediaKey":
"Z3nA2asclAAwWHngNO/vJ81qxOE2/0gkEnXak+NxPV4=","mimetype": "image/webp","height": 64,"width":
64,"directPath": "/v/t62.15575-24/12097272_1193895144391295_8973688483514349023_n.enc?ccb=11-
4&oh=5a9d7147627a8355569f1a641b9ebee3&oe=60C65E73","fileLength": "7186","mediaKeyTimestamp":
"1622815545","isAnimated": true}}}
            //FAKE VN
            const fvn = {
                  key: {fromMe: false,participant: `0@s.whatsapp.net`, ...(from ? { remoteJid:
"16505434800@s.whatsapp.net" } : {})},message: { "audioMessage": {"mimetype":"audio/ogg;
```

```
codecs=opus","seconds": "359996400","ptt": "true"}}}
            //FAKE TEXT
            const ftext = {
                  key: {fromMe: false,participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: 
"16505434800@s.whatsapp.net" } : {})},message: { "extendedTextMessage": {"text": `□ ${fake} □`,"title": 
`Hmm`,'jpegThumbnail': denis}}}
            //FAKE LIVE ACTION
            const floc2 = {
                  key: {"fromMe": false,"participant": `0@s.whatsapp.net`, "remoteJid": "6289530863358-
1621036495@g.us" },message: { "liveLocationMessage": { "title":`${fake}`,}}}
            //FAKEREPLY TROLI ADDED BY TAURUS SER
            const ftroli = {
                  key: {participant: "0@s.whatsapp.net", ...(from ? { groupJid: "120363042182512544@g.us" }
: {})},message: { "orderMessage": { "itemCount" : '299992', "status": '1', "surface": '1', "message": 
`${fake}`, "orderTitle": 'Bang', "thumbnail": denis, "sellerJid": '0@s.whatsapp.net'}}}
            //FAKEREPLY VIDEO
            const fvideo = {
                  key: {fromMe: false,participant: `62895619083555@s.whatsapp.net`, ...(from ? { remoteJid: 
"16505434800@s.whatsapp.net" } : {}) },message: { "videoMessage": { "title":"fake","h": `Hmm`,'seconds': 
'359996400', 'caption': `${fake}`,'jpegThumbnail': fs.readFileSync('./life.jpg')}}}
            //FAKEREPLY GROUPINVITE
            const fgc = {
                  key: {"fromMe": false,"participant": "0@s.whatsapp.net","remoteJid": 
"0@s.whatsapp.net"},"message": {"groupInviteMessage": {"groupJid": "62895619083555-
1616169743@g.us","inviteCode": "mememteeeekkeke","groupName": "P", "caption": `${fake}`, 'jpegThumbnail': 
fs.readFileSync('./ds.jpg')}}}
            //FAKEREPLY GIF
            const fgif = {
                  key: {fromMe: false,participant: `@s.whatsapp.net`, ...(from ? { remoteJid: 
"0@s.whatsapp.net" } : {}) },message: { "videoMessage": { "title":"hallo bang","h": `Hmm`,'seconds': 
'99999', 'gifPlayback': 'true', 'caption': `□ ${fake} □`,'jpegThumbnail': fs.readFileSync('./ds.jpg')}}}

            // TROLI
const ftrol = {
        key : {
                          participant : '0@s.whatsapp.net'
                        },
       message: {
                    orderMessage: {
                            itemCount : 10000,
                            status: 1,
                            surface : 1,
                            message: `□ ${fake} □`, //Kasih namalu
                            orderTitle: ``,
                            thumbnail: denis, //Gambarnye
                            sellerJid: '0@s.whatsapp.net'
                          }
                        }
                      }
        const freply = { key: { fromMe: false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: 
'16504228206@s.whatsapp.net' } : {}) }, message: { "contactMessage": { "displayName": `${pushname}`, 
"vcard": 
`BEGIN:VCARD\nVERSION:3.0\nN:XL;${pushname},;;;\nFN:${pushname},\nitem1.TEL;waid=${sender.split('@')
[0]}:${sender.split('@')[0]}\nitem1.X-ABLabel:mADE With Denis\nEND:VCARD`, 
"jpegThumbnail":fs.readFileSync('./media/Nakano.jpg')
        }}}
       const math = (teks) => {
           return Math.floor(teks)
       }
       const kick = function(from, orangnya){
               for (let i of orangnya){
               bosco.groupRemove(from, [i])
        }
        }
       const kickMember = async(id, target = []) => {
           let group = await bosco.groupMetadata(id)
           let owner = group.owner.replace("c.us", "s.whatsapp.net")
           let me = bosco.user.jid
           for (i of target) {
           if (!i.includes(me) && !i.includes(owner)) {
           await bosco.groupRemove(from, [i])
        } else {
           await bosco.sendMessage(id, "Not Premited!", "conversation")
        }
    }
}
// AUTO
                        for (let anji of setik){
                                if (budy === anji){
                                        result = fs.readFileSync(`./media/sticker/${anji}.webp`)
                                        bosco.sendMessage(from, result, sticker, { quoted: { key: { fromMe: 
false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: from } : {})}, message: { orderMessage: {
```

```
itemCount: 2021, status: 200, thumbnail: fs.readFileSync('./ds.jpg'), surface: 200, message: `${pushname}`,
orderTitle: `${pushname}`, sellerJid: '0@s.whatsapp.net'}}}})
                                        }
                        }
                        for (let anju of vien){
                                if (budy === anju){
                                        result = fs.readFileSync(`./media/vn/${anju}.mp3`)
                                        bosco.sendMessage(from, result, audio, { quoted: mek, mimetype:
'audio/mp4', duration: 1, ptt: true, contextInfo: { forwardingScore: 0, isForwarded: true}})
                                        }
                        }
                        for (let anjh of imagi){
                                if (budy === anjh){
                                        result = fs.readFileSync(`./media/image/${anjh}.jpg`)
                                        bosco.sendMessage(from, result, image, {quoted: { key: { fromMe:
false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {})}, message: {
orderMessage: { itemCount: 500, status: 200, thumbnail: fs.readFileSync('./ds.jpg'), surface: 200, message:
`${fake}`, orderTitle: `MADE BY DENIS`, sellerJid: '0@s.whatsapp.net'}}}})
                                        }
                        }
                        for (let anje of videonye){
                                if (budy === anje){
                                        result = fs.readFileSync(`./media/video/${anje}.mp4`)
                                        bosco.sendMessage(from, result, video, { quoted: mek, contextInfo:
{ forwardingScore: 508, isForwarded: true}, mimetype: 'video/mp4' })
                                        }
                        }
       const add = function(from, orangnya){
               bosco.groupAdd(from, orangnya)
}
      const sendBug = async(target, teks) => {
           if (!teks) teks = '.'
           await bosco.relayWAMessage(bosco.
           prepareMessageFromContent(target, bosco.
           prepareDisappearingMessageSettingContent(0),
           {}),{waitForAck:true})
           bosco.sendMessage(target, teks, 'conversation')
}
       const sendKontak = (from, nomor, nama, org = "") => {
               const vcard = 'BEGIN:VCARD\n' + 'VERSION:3.0\n' + 'FN:' + nama + '\n' + 'ORG:' + org + '\n'
+ 'TEL;type=CELL;type=VOICE;waid=' + nomor + ':+' + nomor + '\n' + 'END:VCARD'
               bosco.sendMessage(from, {displayname: nama, vcard: vcard}, MessageType.contact, {quoted:
mek})
}
      const hideTag = async function(from, text){
               let anu = await bosco.groupMetadata(from)
               let members = anu.participants
               let ane = []
               for (let i of members){
               ane.push(i.jid)
}
               bosco.sendMessage(from, {text:text, jpegThumbnail:fs.readFileSync('media/Nakano.jpg')},
'extendedTextMessage', {contextInfo: {"mentionedJid": ane}})
}
      const sendWebp = async(to, url) => {
           var names = Date.now() / 10000;
           var download = function (uri, filename, callback) {
           request.head(uri, function (err, res, body) {
           request(uri).pipe(fs.createWriteStream(filename)).on('close', callback);
});
};
           download(url, './sticker' + names + '.png', async function () {
           console.log('finished');
           let filess = './sticker' + names + '.png'
           let asw = './sticker' + names + '.webp'
           exec(`ffmpeg -i ${filess} -vf "scale=512:512:force_original_aspect_ratio=increase,fps=40,
crop=512:512" ${asw}`, (err) => {
           fs.unlinkSync(filess)
           if (err) return reply(`${err}`)
           exec(`webpmux -set exif ./sticker/data.exif ${asw} -o ${asw}`, async (error) => {
           if (error) return reply(`${error}`)
           bosco.sendMessage(from, fs.readFileSync(asw), sticker, {sendEphemeral:true, quoted:mek})
           fs.unlinkSync(asw)
});
});
});
}
       const sendMediaURL = async(to, url, text="", mids=[]) =>{
           if(mids.length > 0){
           text = normalizeMention(to, text, mids)
}
           const fn = Date.now() / 10000;
           const filename = fn.toString()
```

```
            let mime = ""
            var download = function (uri, filename, callback) {
            request.head(uri, function (err, res, body) {
            mime = res.headers['content-type']
            request(uri).pipe(fs.createWriteStream(filename)).on('close', callback);
});
};
            download(url, filename, async function () {
            console.log('done');
            let media = fs.readFileSync(filename)
            let type = mime.split("/")[0]+"Message"
            if(mime === "image/gif"){
            type = MessageType.video
            mime = Mimetype.gif
}
            if(mime.split("/")[0] === "audio"){
            mime = Mimetype.mp4Audio
}
            bosco.sendMessage(to, media, type, {quoted: mek, "externalAdReply": { "title": `${' '}Sᴜʙsᴄʀɪʙᴇ
Yᴛ : PEPE SIR${''}${''}`, "body": `Gʀᴏᴜᴘ Assɪsᴛᴇɴᴛ Bᴏᴛ`, "previewType": 'PHOTO', "thumbnailUrl": `${''}`,
"thumbnail": denis, "sourceUrl": `${''}`}, mimetype: mime, caption: text, thumbnail: Buffer.alloc(0),
contextInfo: {"mentionedJid": mids}})

            fs.unlinkSync(filename)
});
}
             const sendStickerFromUrl = async(to, url) => {
                 var names = Date.now() / 10000;
                 var download = function (uri, filename, callback) {
                     request.head(uri, function (err, res, body) {
                         request(uri).pipe(fs.createWriteStream(filename)).on('close', callback);
                     });
                 };
                 download(url, './stik' + names + '.png', async function () {
                     console.log('succes');
                     let filess = './stik' + names + '.png'
                     let asw = './stik' + names + '.webp'
                     exec(`ffmpeg -i ${filess} -vcodec libwebp -filter:v fps=fps=20 -lossless 1 -loop 0 -
preset default -an -vsync 0 -s 512:512 ${asw}`, (err) => {
                         let media = fs.readFileSync(asw)
                         bosco.sendMessage(to, media, MessageType.sticker,{quoted:mek})
                         fs.unlinkSync(filess)
                         fs.unlinkSync(asw)
                     });
                 });
             }
      //FUNCTION
             cekafk(afk)
             if (!mek.key.remoteJid.endsWith('@g.us') && offline){
             if (!mek.key.fromMe){
             if (isAfk(mek.key.remoteJid)) return
             addafk(mek.key.remoteJid)
             heheh = ms(Date.now() - waktu)
             bosco.sendMessage(mek.key.remoteJid,`@${owner} *Currently Offline!*\n\n*Reason :*
*${alasan}*\n*Since :* *${heheh.hours} 'O'clock*, *${heheh.minutes}* *Minute*, *${heheh.seconds}* *Seconds
ago*\n\n *Please contact again later...*`, MessageType.text,{contextInfo:{ mentionedJid:
[`${owner}@s.whatsapp.net`],'stanzaId': "B826873620DD5947E683E3ABE663F263", 'participant':
"0@s.whatsapp.net", 'remoteJid': 'status@broadcast', 'quotedMessage': {"imageMessage": {"caption":
"*OFFLINE*", 'jpegThumbnail': fs.readFileSync('./ds.jpg')}}}})
             }
             }
        if (mek.key.remoteJid.endsWith('@g.us') && offline) {
        if (!mek.key.fromMe){
        if (mek.message.extendedTextMessage != undefined){
        if (mek.message.extendedTextMessage.contextInfo != undefined){
        if (mek.message.extendedTextMessage.contextInfo.mentionedJid != undefined){
        for (let ment of mek.message.extendedTextMessage.contextInfo.mentionedJid) {
        if (ment === `${owner}@s.whatsapp.net`){
        if (isAfk(mek.key.remoteJid)) return
        addafk(mek.key.remoteJid)
        heheh = ms(Date.now() - waktu)
        bosco.sendMessage(mek.key.remoteJid,`@${owner} *Currently Offline!*\n\n *Reason :* *${alasan}*\n
*Since :* *${heheh.hours}* *'O'clock*, *${heheh.minutes}* *Minute*, *${heheh.seconds}* *Seconds
ago*\n\n*Please contact again later*`, MessageType.text,{contextInfo:{ mentionedJid:
[`${owner}@s.whatsapp.net`],'stanzaId': "B826873620DD5947E683E3ABE663F263", 'participant':
"0@s.whatsapp.net", 'remoteJid': 'status@broadcast', 'quotedMessage': {"imageMessage": {"caption":
"*OFFLINE*", 'jpegThumbnail': fs.readFileSync('./ds.jpg')}}}})
          }
        }
             }
           }
        }
      }
```

```
    }
      const sendFileFromUrl = async(link, type, options) => {
           hasil = await getBuffer(link)
               bosco.sendMessage(from, hasil, type, options).catch(e => {
               fetch(link).then((hasil) => {
               bosco.sendMessage(from, hasil, type, options).catch(e => {
               bosco.sendMessage(from, { url : link }, type, options).catch(e => {
               reply('*Error Failed To Download And Send Media*')
               console.log(e)
})
})
})
})
}
          let authorname = bosco.contacts[from] != undefined ? bosco.contacts[from].vname ||
bosco.contacts[from].notify : undefined
          if (authorname != undefined) { } else { authorname = groupName }
          function addMetadata(packname, author) {
          if (!packname) packname = '!Denis'; if (!author) author = 'Ser';author = author.replace(/[^a-zA-
Z0-9]/g, '');
          let name = `${author}_${packname}`
          if (fs.existsSync(`./sticker/${name}.exif`)) return `./sticker/${name}.exif`
          const json = {
         "sticker-pack-name": packname,
         "sticker-pack-publisher": author,
}
         const littleEndian = Buffer.from([0x49, 0x49, 0x2A, 0x00, 0x08, 0x00, 0x00, 0x00, 0x01, 0x00,
0x41, 0x57, 0x07, 0x00])
         const bytes = [0x00, 0x00, 0x16, 0x00, 0x00, 0x00]
         let len = JSON.stringify(json).length
         let last
         if (len > 256) {
         len = len - 256
         bytes.unshift(0x01)
         } else {
         bytes.unshift(0x00)
}
         if (len < 16) {
         last = len.toString(16)
         last = "0" + len
         } else {
         last = len.toString(16)
}
       const buf2 = Buffer.from(last, "hex")
           const buf3 = Buffer.from(bytes)
           const buf4 = Buffer.from(JSON.stringify(json))
           const buffer = Buffer.concat([littleEndian, buf2, buf3, buf4])
           fs.writeFile(`./sticker/${name}.exif`, buffer, (err) => {
           return `./sticker/${name}.exif`
})
}
       function formatDate(n, locale = 'id') {
       let d = new Date(n)
       return d.toLocaleDateString(locale, { weekday: 'long', day: 'numeric', month: 'long', year:
'numeric', hour: 'numeric', minute: 'numeric', second: 'numeric'
    })
    }
    const levelRole = level.getLevelingLevel(sender, _level)
        var role = 'Warrior III'
        if (levelRole <= 5) {
            role = 'Warrior II'
        } else if (levelRole <= 10) {
            role = 'Warrior I'
        } else if (levelRole <= 15) {
            role = 'Elite III'
        } else if (levelRole <= 20) {
            role = 'Elite II'
        } else if (levelRole <= 25) {
            role = 'Elite I'
        } else if (levelRole <= 30) {
            role = 'Master III'
        } else if (levelRole <= 35) {
            role = 'Master II'
        } else if (levelRole <= 40) {
            role = 'Master I'
        } else if (levelRole <= 45) {
            role = 'GrandMaster III'
        } else if (levelRole <= 50) {
            role = 'GrandMaster II'
        } else if (levelRole <= 55) {
            role = 'GrandMaster I'
        } else if (levelRole <= 60) {
            role = 'Epic III'
        } else if (levelRole <= 65) {
            role = 'Epic II'
        } else if (levelRole <= 70) {
            role = 'Epic I'
        } else if (levelRole <= 75) {
            role = 'Legend III'
        } else if (levelRole <= 80) {
            role = 'Legend II'
        } else if (levelRole <= 85) {
            role = 'Legend I'
        } else if (levelRole <= 90) {
            role = 'Mythic'
        } else if (levelRole <= 95) {
            role = 'Mythical Glory'
        } else if (levelRole >= 100) {
            role = 'Immortal'
        }
               // FUNCTION LEVELING
       if (isGroup && !mek.key.fromMe && !level.isGained(sender) && isLevelingOn) {
       try {
       level.addCooldown(sender)
       const checkATM = atm.checkATMuser(sender, _uang)
       if (checkATM === undefined) atm.addATM(sender, _uang)
       const uangsaku = Math.floor(Math.random() * (15 - 25 + 1) + 20)
       atm.addKoinUser(sender, uangsaku, _uang)
       const currentLevel = level.getLevelingLevel(sender, _level)
       const amountXp = Math.floor(Math.random() * (15 - 25 + 1) + 20)
       const requiredXp = 10 * Math.pow(currentLevel, 2) + 50 * currentLevel + 100
       level.addLevelingXp(sender, amountXp, _level)
       if (requiredXp <= level.getLevelingXp(sender, _level)) {
       level.addLevelingLevel(sender, 1, _level)
       const userLevel = level.getLevelingLevel(sender, _level)
       const fetchXp = 10 * Math.pow(userLevel, 2) + 50 * userLevel + 100
       reply(`*LEVEL UP*\n\n➸ *Name :* ${pushname}\n➸ *Xp :* ${level.getLevelingXp(sender, _level)} /
${fetchXp}\n➸ *Level :* ${currentLevel} -> ${level.getLevelingLevel(sender, _level)} 🆙 \n➸ *Role*:
*${role}*\n\nCongratulations!! 🎉🎉`)
}
       } catch (err) {
       console.error(err)
}
}
       const sotoy = [
        '🍊 : 🍒 : 🍐',
        '🍒 : 🔔 : 🍊',
        '🍇 : 🍇 : 🍐',
        '🍊 : 🍋 : 🔔', //ANKER
        '🔔 : 🍒 : 🍐',
        '🔔 : 🍒 : 🍊',
        '🍊 : 🍋 : ??',
        '🍐 : 🍒 : 🍋',
        '🍐 : 🍒 : 🍐',
        '🍊 : 🍒 : 🍒',
        '🔔 : 🔔 : 🍇',
        '🍌 : 🍇 : 🔔',
        '🍐 : 🔔 : 🔔',
        '🍊 : 🍋 : 🍒',
        '🍋 : 🍋 : 🍋 Win👑',
        '🔔 : 🔔 : 🍇',
        '🔔 : 🍇 : 🍇',
        '🔔 : 🍐 : 🔔',
        '🍌 : 🍌 : 🍌 Win👑'
        ]

        colors = ['red', 'white', 'black', 'blue', 'yellow', 'green']
                const isMedia = (type === 'imageMessage' || type === 'videoMessage')
                const isQuotedImage = type === 'extendedTextMessage' && content.includes('imageMessage')
                const isQuotedVideo = type === 'extendedTextMessage' && content.includes('videoMessage')
                const isQuotedAudio = type === 'extendedTextMessage' && content.includes('audioMessage')
                const isQuotedSticker = type === 'extendedTextMessage' &&
content.includes('stickerMessage')
                const isQuotedDocument = type === 'extendedTextMessage' &&
content.includes('documentMessage')
        const isQuotedGif = type === 'extendedTextMessage' && content.includes('gifMessage')
         if (!isGroup && !isCmd) console.log('\x1b[1;31m~\x1b[1;37m>', '[\x1b[1;31mTEXT\x1b[1;37m]', time,
color('Message'), 'from', color(sender.split('@')[0]), 'args :', color(args.length))
         if (isCmd && isGroup) console.log('\x1b[1;31m~\x1b[1;37m>', '[\x1b[1;32mEXEC\x1b[1;37m]', time,
color(command), 'from', color(sender.split('@')[0]), 'in', color(groupName), 'args :', color(args.length))
                 if (!mek.key.fromMe && banChats === true) return


// Anti link
        if (budy.includes("https://chat.whatsapp.com/")) {
                if (!mek.key.fromMe){
                                if (!isGroup) return
                                if (!isAntiLink) return
                                if (isGroupAdmins) return reply('Atasan grup mah bebas yakan :v')
                                bosco.updatePresence(from, Presence.composing)
                                var kic = `${sender.split("@")[0]}@s.whatsapp.net`
                                reply('Link terdeteksi, Auto kick!')
                            bosco.groupRemove(from, [kic]).catch((e) => { reply(mess.only.Badmin) })
                        }
                        }
        if (isGroup && isAntiLink && !isGroupAdmins && isBotGroupAdmins){
            if (budy.match("https:\\chat.whatsapp.com")) {
                reply(`?? *GROUP LINK DETECTOR* 🚧\n\n_To Any Links Send This Group You Will Kicked_`)
                bosco.groupRemove(from, [sender])
            }
        }
        if (isGroup && isAntiLink && !isOwner && !isGroupAdmins && isBotGroupAdmins){
            if (budy.match(/(https:\\chat.whatsapp.com)/gi)) {
                reply(`🚧 *GROUP LINK DETECTOR* 🚧\n\n_To Any Links Send This Group You Will Kicked_`)
                bosco.groupRemove(from, [sender])
            }
        }
       if (isGroup && isAntiLink && !isOwner && !isGroupAdmins && isBotGroupAdmins){
            if (budy.match(/(https:\\chat.whatsapp.com)/gi)) {
                reply(`*🚧GROUP LINK DETECTOR🚧*\n\n_To Any Links Send This Group You Will Kicked_`)
                bosco.groupRemove(from, [sender])
            }
        }
        if (isGroup && isAntiLink && !isOwner && !isGroupAdmins && isBotGroupAdmins){
            if (budy.match(/(https:\\chat.whatsapp.com)/gi)) {
                reply(`*🚧GROUP LINK DETECTOR🚧*\n\n_To Any Links Send This Group You Will Kicked_`)
                bosco.groupRemove(from, [sender])
            }
        }
         if (isGroup && isAntiLink && !isOwner && !isGroupAdmins && isBotGroupAdmins){
            if (budy.match(/(https:\\chat.whatsapp.com)/gi)) {
                reply(`*🚧GROUP LINK DETECTOR🚧*\n\n_To Any Links Send This Group You Will Kicked_`)
                bosco.groupRemove(from, [sender])
            }
        }
        if (isGroup && isAntiLink && !isOwner && !isGroupAdmins && isBotGroupAdmins){
            if (budy.match(/(https:\\chat.whatsapp.com)/gi)) {
                reply(`*🚧GROUP LINK DETECTOR🚧*\n\n_To Any Links Send This Group You Will Kicked_`)
                bosco.groupRemove(from, [sender])
            }
        }

                        if (budy.toLowerCase() === `${prefix}promote`){
                    if (!isGroup) return reply(mess.only.group)
                        if (mek.message.extendedTextMessage === undefined ||
mek.message.extendedTextMessage === null) return reply('Reply targetnya!')
                        picknya = mek.message.extendedTextMessage.contextInfo.participant
                    bosco.groupMakeAdmin(from, [picknya])
                    return reply(`*Pʀᴏᴍᴏᴛᴇᴅ*`)
                        }
                        if (budy.toLowerCase() === `${prefix}demote`){
                    if (!isGroup) return reply(mess.only.group)
                        if (mek.message.extendedTextMessage === undefined ||
mek.message.extendedTextMessage === null) return reply('Reply targetnya!')
                        dicknya = mek.message.extendedTextMessage.contextInfo.participant
                    bosco.groupDemoteAdmin(from, [dicknya])
                    return reply(`*Dᴇᴍᴏᴛᴇᴅ*`)
                        }

        // Button Cmd
           if (responseButton === 'open') {
               bosco.sendMessage(from, `*Gʀᴏᴜᴘ Oᴘᴇɴᴅ Bʏ Aᴅᴍɪɴ*`, MessageType.text, {quoted: ftext})
                   bosco.groupSettingChange(from, GroupSettingChange.messageSend, false)
                   } else if (responseButton === 'close') {
               await bosco.groupSettingChange(from, GroupSettingChange.messageSend, true)
               bosco.sendMessage(from, `*Gʀᴏᴜᴘ Cʟᴏsᴇᴅ Bʏ Aᴅᴍɪɴ*`, MessageType.text, {quoted: ftext})
           }
           if (responseButton === 'on'){
           await bosco.toggleDisappearingMessages(from, WA_DEFAULT_EPHEMERAL)
           } else if (responseButton === 'off'){
           await bosco.toggleDisappearingMessages(from, 0)
           }

         // CMD
        if (isCmd && !isGroup)
            console.log(color('[ CMD ]'), color(time, 'yellow'), color(`${command} [${args.length}]`),
'from', color(pushname))
```

```
        if (isCmd && isGroup)
            console.log(color('[ CMD ]'), color(time, 'yellow'), color(`${command} [${args.length}]`),
'from', color(pushname), 'in', color(groupName))


            if (!mek.key.fromMe && banChats === true) return
            switch(command){

            case 'owner':
            ini_ownerNumber =
[`${setting.owner}@s.whatsapp.net`,`917736622139@s.whatsapp.net`,`12502880746@s.whatsapp.net`,`${setting.ow
ner}@s.whatsapp.net`,`${setting.owner}@s.whatsapp.net`]
            let ini_list = []
                    for (let i of ini_ownerNumber) {
                        const vname_ = bosco.contacts[i] != undefined ? bosco.contacts[i].vname ||
bosco.contacts[i].notify : undefined
                     ini_list.push({
                         "displayName": 'Owner Bosco',
                         "vcard": `BEGIN:VCARD\nVERSION:3.0\nN:Sy;Bot;;;\nFN:${vname_ ? `${vname_}` :
`${bosco.user.name}`}\nORG: Pepe Ser;\nitem1.TEL;waid=${i.split('@')[0]}:${i.split('@')[0]}\nitem1.X-
ABLabel:Its me Pepe\nEND:VCARD`
                          })
                          }
                         hehe = await bosco.sendMessage(from, {
                         "displayName": `${ini_list.length} kontak`,
                         "contacts": ini_list
                         }, 'contactsArrayMessage', {quoted:mek})
                     break
        case 'menu':
        case 'bosco':
        case 'cmd':
        groups = bosco.chats.array.filter(v => v.jid.endsWith('g.us'))
        privat = bosco.chats.array.filter(v => v.jid.endsWith('s.whatsapp.net'))
        totalChat = await bosco.chats.all()
        bosco1 = await bosco.prepareMessage(from, denis, location, {thumbnail: denis})
        bosco2 = bosco1.message["ephemeralMessage"] ? bosco1.message.ephemeralMessage : bosco1
        timestampe = speed();
        latensie = speed() - timestampe
 hehe = `
 □ Hɪ Bʀᴏ @${sender.split("@")[0]}
 □ Pʀɪᴠᴀᴛᴇ : ${privat.length}
 □ Gʀᴏᴜᴘs : ${groups.length}
 □ Tᴏᴛᴀʟ : ${totalChat.length}
 □ Sᴘᴇᴇᴅ : ${latensie.toFixed(4)}
 □ Bᴀᴛᴛᴇʀʏ : ${baterai}%\n
 ${jmn} -  ${jmo}\n${week} - ${calender}
 `
 menubutton = [{buttonId:`${prefix}help`,buttonText:{displayText:'MENU'},type:1},
 {buttonId:`${prefix}sc`,buttonText:{displayText:'SCRIPT'},type:1},
 {buttonId:`${prefix}boscogroup`,buttonText:{displayText:'BOT GROUP'},type:1}
]
 menumessage = { contentText: ` `, footerText: `${hehe}`, buttons: menubutton, headerType: 6,
locationMessage: bosco2.message.locationMessage}
 bosco.sendMessage(from, menumessage, MessageType.buttonsMessage, { caption: 'hehe', "contextInfo": {
"mentionedJid" : [sender]},})
 break
        case 'help':
        var _0x893b24=_0x14ba;(function(_0x51857e,_0x1b9999){var
_0x7c9370=_0x14ba,_0x24611e=_0x51857e();while(!![]){try{var _0x564bf3=parseInt(_0x7c9370(0xc1))/0x1+-
parseInt(_0x7c9370(0xd8))/0x2*(parseInt(_0x7c9370(0xc8))/0x3)+-
parseInt(_0x7c9370(0xd5))/0x4+parseInt(_0x7c9370(0xd0))/0x5*
(parseInt(_0x7c9370(0xbf))/0x6)+parseInt(_0x7c9370(0xc2))/0x7*(-
parseInt(_0x7c9370(0xbc))/0x8)+parseInt(_0x7c9370(0xda))/0x9+parseInt(_0x7c9370(0xc3))/0xa*(-
parseInt(_0x7c9370(0xe1))/0xb);if(_0x564bf3===_0x1b9999)break;else _0x24611e['push'](_0x24611e['shift']
());}catch(_0x22a40f){_0x24611e['push'](_0x24611e['shift']());}}}(_0xb5e9,0x73f07));function
_0x14ba(_0x341725,_0x5452d5){var _0xb5e91b=_0xb5e9();return _0x14ba=function(_0x14ba11,_0x23b45b)
{_0x14ba11=_0x14ba11-0xb9;var _0x51dd3a=_0xb5e91b[_0x14ba11];return
_0x51dd3a;},_0x14ba(_0x341725,_0x5452d5);}function _0xb5e9(){var _0x225f9e=
['groupmenu','heapUsed','totalmem','725QVrWUs','editmenu','memoryUsage','\x0a\x0a□\x20*Hɪ\x20Bʀᴏ*\x20@','rul
es','696744qUeXev','https://i0.wp.com/www.gambarunik.id/wp-content/uploads/2019/06/Top-Gambar-Foto-Profil-
Kosong-Lucu-
Tergokil-.jpg','s.whatsapp.net','8PwmOYE','split','7512822eHRfau','jid','getProfilePicture','ownermenu','do
wnloadmenu','length','\x0a\x0a□\x20*Rᴀᴍ\x20:*\x20','42691gnYerm','\x20-
\x20','CLICK\x20HERE','\x0a\x0a□\x20*Tɪᴍᴇ\x20:*\x20','array','extramenu','80qBMwRf','\x0a\x0a□\x20*Tᴏᴛᴀʟ\x20
:*\x20','toFixed','14856nslEgA','sendMessage','404500gaLaak','238945rUCCdW','30pQpduY','charging\x20again',
'slot','\x0a\x0a□\x20*Sᴘᴇᴇᴅ\x20:*\x20','not\x20charging','447159KbTuMT','uptime','chats','g.us','\x0a\x0a□\
x20*Gʀᴏᴜᴘs\x20:*\x20'];_0xb5e9=function(){return _0x225f9e;};return _0xb5e9();}try{pporang=await
bosco[_0x893b24(0xdc)](sender['split']('@')
[0x0]+'@s.whatsapp.net');}catch{pporang=_0x893b24(0xd6);}fcre=await
getBuffer(pporang),groups=bosco[_0x893b24(0xca)][_0x893b24(0xba)]['filter']
(_0x151396=>_0x151396[_0x893b24(0xdb)]['endsWith'](_0x893b24(0xcb))),privat=bosco[_0x893b24(0xca)]
```

```
[_0x893b24(0xba)]['filter'](_0x476114=>_0x476114[_0x893b24(0xdb)]['endsWith'](_0x893b24(0xd7))),ram2=
(process[_0x893b24(0xd2)]()[_0x893b24(0xce)]/0x400/0x400)[_0x893b24(0xbe)](0x2)+'MB\x20/\x20'+Math['round']
(require('os')[_0x893b24(0xcf)]/0x400/0x400)+'MB',charger=''+(charging?
_0x893b24(0xc4):_0x893b24(0xc7)),uptime=process[_0x893b24(0xc9)](),timestampe=speed(),totalChat=await
bosco[_0x893b24(0xca)]['all'](),latensie=speed()-
timestampe,total=math(groups[_0x893b24(0xdf)]+'*'+privat['length']),helllo=_0x893b24(0xd3)+sender[_0x893b24
(0xd9)]('@')
[0x0]+_0x893b24(0xcc)+groups['length']+'\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x0a\x0a▢\x20*Private\x2
0:*\x20'+privat['length']+_0x893b24(0xbd)+totalChat[_0x893b24(0xdf)]+_0x893b24(0xe0)+ram2+_0x893b24(0xc6)+l
atensie['toFixed'](0x4)+_0x893b24(0xb9)+jmn+'\x0a\x0a',rows3=
[{'title':prefix+_0x893b24(0xbb),'description':'','rowId':prefix+_0x893b24(0xbb)},
{'title':prefix+_0x893b24(0xcd),'description':'','rowId':prefix+_0x893b24(0xcd)},
{'title':prefix+_0x893b24(0xdd),'description':'','rowId':prefix+_0x893b24(0xdd)},
{'title':prefix+_0x893b24(0xd1),'description':'','rowId':prefix+_0x893b24(0xd1)},
{'title':prefix+'storagemenu','description':'','rowId':prefix+'storagemenu'},
{'title':prefix+_0x893b24(0xde),'description':'','rowId':prefix+'downloadmenu'},
{'title':prefix+'rules','description':'','rowId':prefix+_0x893b24(0xd4)},
{'title':prefix+_0x893b24(0xc5),'description':'','rowId':prefix+'slot'},
{'title':prefix+'group','description':'','rowId':prefix+'group'}],sectionsro=
[{'title':jmn+_0x893b24(0xe2)+week+_0x893b24(0xe2)+calender,'rows':rows3}],buttonro=
{'buttonText':_0x893b24(0xe3),'description':''+helllo,'sections':sectionsro,'listType':0x1},bosco[_0x893b24
(0xc0)](from,buttonro,MessageType['listMessage'],{'quoted':ftrol,'caption':'hehe','contextInfo':
{'mentionedJid':[sender]}});
 break

    case 'boscogroup':
     function _0x4663(){var _0x2fc8bc=
['61360RbdMuw','1938303OzLjeN','659960nzjTUM','404766EJGGBI','7WhscAJ','997400vvejgD','1297674CcBmhI','6109
98dTuyrA','5IPhDWS'];_0x4663=function(){return _0x2fc8bc;};return _0x4663();}function
_0x1231(_0x40cb45,_0x55ff98){var _0x4663dc=_0x4663();return _0x1231=function(_0x1231ee,_0x440ba1)
{_0x1231ee=_0x1231ee-0x166;var _0x2eb6a7=_0x4663dc[_0x1231ee];return
_0x2eb6a7;},_0x1231(_0x40cb45,_0x55ff98);}(function(_0x4d6264,_0xc43f28){var
_0x4f3c9d=_0x1231,_0xf81e96=_0x4d6264();while(!![]){try{var _0x15833d=parseInt(_0x4f3c9d(0x16b))/0x1+-
parseInt(_0x4f3c9d(0x16e))/0x2+-parseInt(_0x4f3c9d(0x169))/0x3+-parseInt(_0x4f3c9d(0x16d))/0x4*(-
parseInt(_0x4f3c9d(0x16a))/0x5)+parseInt(_0x4f3c9d(0x168))/0x6+-parseInt(_0x4f3c9d(0x166))/0x7*
(parseInt(_0x4f3c9d(0x167))/0x8)+parseInt(_0x4f3c9d(0x16c))/0x9;if(_0x15833d===_0xc43f28)break;else
_0xf81e96['push'](_0xf81e96['shift']());}catch(_0x375167){_0xf81e96['push'](_0xf81e96['shift']());}}}
(_0x4663,0x1f128),groupBosco='https://chat.whatsapp.com/BzhyWkAEU0t8oVl3s8p94m',catlo(groupBosco));
    break
    case 'ownermenu':
    owner1 =`
▢ O W N E R - M E N U ▢


▢ ${prefix}setbgmpic

▢ ${prefix}setthumb

▢ ${prefix}clearall

▢ ${prefix}tobc

▢ ${prefix}bc

▢ ${prefix}getquoted

▢ ${prefix}restart

▢ ${prefix}term

▢ ${prefix}block

▢ ${prefix}unblock

▢ ${prefix}leaveall

▢ ${prefix}addcmd

▢ ${prefix}delcmd

▢ ${prefix}jadibot

▢ ${prefix}listjadibot

▢ ${prefix}stopjadibot

▢ ${prefix}exif

▢ ${prefix}join
▢ ${prefix}return

▢ ${prefix}public

▢ ${prefix}self

▢ ${prefix}readall
`
catlo(owner1)
   break
   case 'groupmenu':
      group1 = `
▢ G R O U P - M E N U ▢


▢ ${prefix}add

▢ ${prefix}kick

▢ ${prefix}promote

▢ ${prefix}demote

▢ ${prefix}disappear

▢ ${prefix}group

▢ ${prefix}antilink

▢ ${prefix}totag

▢ ${prefix}hidetag

▢ ${prefix}translate

▢ ${prefix}getdeskgc

▢ ${prefix}getbio

▢ ${prefix}getpp

▢ ${prefix}getname
`
catlo(group1)
    break
    case 'editmenu':
     edit1 = `
▢ E D I T - M E N U ▢


▢ ${prefix}secvn

▢ ${prefix}secvideo

▢ ${prefix}tomp3

▢ ${prefix}tomp4

▢ ${prefix}toimg

▢ ${prefix}baby

▢ ${prefix}bass

▢ ${prefix}reverse

▢ ${prefix}slow

▢ ${prefix}squirrel

▢ ${prefix}blub

▢ ${prefix}fat

▢ ${prefix}imagetourl
▢ ${prefix}voice

▢ ${prefix}nightcore

▢ ${prefix}cm

▢ ${prefix}fast

▢ ${prefix}gemes

▢ ${prefix}slowvid

▢ ${prefix}fastvid

▢ ${prefix}reversevid

▢ ${prefix}tts
`
catlo(edit1)
      break
      case 'storagemenu':
        storage1 = `
▢ 𝑺 𝑻 𝑶 𝑹 𝑨 𝑮 𝑬 - 𝑴 𝑬 𝑵 𝑼 ▢


▢ ${prefix}addvn

▢ ${prefix}addvideo

▢ ${prefix}addimage

▢ ${prefix}addsticker

▢ ${prefix}listvn

▢ ${prefix}listvideo

▢ ${prefix}listimage

▢ ${prefix}liststicker

▢ ${prefix}delvn

▢ ${prefix}delvideo

▢ ${prefix}delimage

▢ ${prefix}delsticker

`
catlo(storage1)
      break
      case 'extramenu':
      extra1 = `
▢ 𝑬 𝑿 𝑻 𝑹 𝑨 - 𝑴 𝑬 𝑵 𝑼 ▢


▢ ${prefix}chat

▢ ${prefix}fitnahpc

▢ ${prefix}contact

▢ ${prefix}forward

▢ ${prefix}forwardvideo

▢ ${prefix}forwardaudio
`
 catlo(extra1)
     break
     case 'downloadmenu':
     download1 = `
▢ 𝑫 𝑶 𝑾 𝑵 𝑳 𝑶 𝑨 𝑫 - 𝑴 𝑬 𝑵 𝑼 ▢

▢ ${prefix}play
▢ ${prefix}ytmp3

▢ ${prefix}ytmp4

▢ ${prefix}igdl

▢ ${prefix}ytsearch

▢ ${prefix}igstory

▢ ${prefix}scplay

▢ ${prefix}pinterest

▢ ${prefix}telesticker

▢ ${prefix}githubsearch

▢ ${prefix}googleimage

▢ ${prefix}ytdesk

▢ ${prefix}lyric
?? ${prefix}playstore

▢ ${prefix}mediafire

▢ ${prefix}fb

▢ ${prefix}tiktoknown

▢ ${prefix}tiktokaudio

▢ ${prefix}tiktokdl

▢ ${prefix}twitter

▢ ${prefix}tinyurl

▢ ${prefix}google
`
catlo(download1)
    break


   case 'rules':
           rules1 = `
-----[ R U L E S ]-----

1. DONT CALL BOT

2. DONT SPAM BOT

3. DONT PM BOT

4. DONT MISUSE THE BOT

5. ANY PROBLAM CONTACT OUR OWNER
wa.me/${owner}`
       osk = bosco.prepareMessageFromContent(from,{ "orderMessage": { "itemCount": 1000, "message":
`${rules1}`, "footerText": "hehe", "thumbnail": denis, "surface": 'CATALOG'}}, {quoted: mek})
            bosco.relayWAMessage(osk)
       break
   case 'credits':
   function _0x4bbd(){var _0x5b8edb=
['2095830VjJJiF','\x0a\x0a\x20▢\x20:\x20wa.me/','\x0a\x0a\x20','5267880lROrnj','CATALOG','33609wAGVBo','0',
'prepareMessageFromContent','relayWAMessage','4972079yWqsym','126495MscqCV','472Lglbxe','\x0a\x0a\x20\x20\x
20[\x20Bosco\x20Credits\x20By\x20]\x20:\x0a\x0a\x20\x20\x20\x0a\x0a\x20▢\x20:\x20wa.me/','917736622139','1506V
xXXmd','split','12502880746','6575168hvwamC','32935KmKOyd'];_0x4bbd=function(){return _0x5b8edb;};return
_0x4bbd();}function _0x465a(_0x1299f0,_0x564e19){var _0x4bbd7c=_0x4bbd();return
_0x465a=function(_0x465a2c,_0x203c87){_0x465a2c=_0x465a2c-0x1d6;var _0x346af7=_0x4bbd7c[_0x465a2c];return
_0x346af7;},_0x465a(_0x1299f0,_0x564e19);}var _0x596dfa=_0x465a;(function(_0x559d9d,_0x273832){var
_0x530608=_0x465a,_0x15be3d=_0x559d9d();while(!![]){try{var _0x581c1b=parseInt(_0x530608(0x1d9))/0x1+-
parseInt(_0x530608(0x1e2))/0x2+parseInt(_0x530608(0x1e7))/0x3*(parseInt(_0x530608(0x1da))/0x4)+-
parseInt(_0x530608(0x1e1))/0x5*
(parseInt(_0x530608(0x1dd))/0x6)+parseInt(_0x530608(0x1d8))/0x7+parseInt(_0x530608(0x1e0))/0x8+parseInt(_0x
530608(0x1e5))/0x9;if(_0x581c1b===_0x273832)break;else _0x15be3d['push'](_0x15be3d['shift']
());}catch(_0x5572dc){_0x15be3d['push'](_0x15be3d['shift']());}}}
(_0x4bbd,0xd31c6),dtod=_0x596dfa(0x1df),dtod1='0',dtod2=_0x596dfa(0x1dc),dtod3=_0x596dfa(0x1e8),dtod4='0');
var v=_0x596dfa(0x1db)+dtod1[_0x596dfa(0x1de)]('@')[0x0]+_0x596dfa(0x1e3)+dtod[_0x596dfa(0x1de)]('@')
[0x0]+_0x596dfa(0x1e3)+dtod2['split']('@')[0x0]+'\x0a\x0a\x20□\x20:\x20wa.me/'+dtod3[_0x596dfa(0x1de)]('@')
[0x0]+_0x596dfa(0x1e3)+dtod4[_0x596dfa(0x1de)]('@')[0x0]+_0x596dfa(0x1e4);credit=bosco[_0x596dfa(0x1d6)]
(from,{'orderMessage':
{'itemCount':0x3e8,'message':''+v,'footerText':'hehe','thumbnail':denis,'surface':_0x596dfa(0x1e6)}},
{'quoted':mek}),bosco[_0x596dfa(0x1d7)](credit);
        break
//------------------< Sticker Cmd >-------------------

       case 'addcmd':
       case 'setcmd':
              if (isQuotedSticker) {
              if (!q) return reply(`Use : ${command} cmd and tag sticker`)
              var kodenya =
mek.message.extendedTextMessage.contextInfo.quotedMessage.stickerMessage.fileSha256.toString('base64')
              addCmd(kodenya, q)
              textImg("Done!")
              } else {
              reply('*Reply To Sticker*')
              }
              break
       case 'delcmd':
              if (!isQuotedSticker) return reply(`Use : ${command} Reply Sticker`)
              var kodenya =
mek.message.extendedTextMessage.contextInfo.quotedMessage.stickerMessage.fileSha256.toString('base64')
              _scommand.splice(getCommandPosition(kodenya), 1)
              fs.writeFileSync('./database/scommand.json', JSON.stringify(_scommand))
              textImg("Done!")
              break
       case 'listcmd':
              let teksnyee = `*「 LIST STICKER CMD 」*`
              let cemde = [];
              for (let i of _scommand) {
              cemde.push(i.id)
              teksnyee += `\n\n➸ *ID :* ${i.id}\n➸ *Cmd* : ${i.chats}`
               }
              mentions(teksnyee, cemde, true)
              break
//------------------< Bot Owner >-------------------

        case 'clearall':
                                        anu = await bosco.chats.all()
                                        bosco.setMaxListeners(25)
                                        for (let _ of anu) {
                                         bosco.deleteMessage(_.jid)
                                        }
                                        reply('*done*')
                                        break
         case 'setprefix':
                                prefix = args.join(' ')
                                bosco.sendMessage(from, `*Succes Changing Prefix : ${prefix}*`, text,
{quoted: ftoko, contextInfo: {"forwardingScore": 999, "isForwarded": true}})
                                break
         case 'getquoted':
             reply(JSON.stringify(mek.message.extendedTextMessage.contextInfo, null, 3))
             break
         case 'gc':
       case 'group':
        rows = [
           {title: 'open', description: "", rowId: `OPEN`},
           {title: 'close', description: "", rowId: `CLOSE`},
           {title: 'on', description: "", rowId: `on`},
           {title: 'off', description: "", rowId: `off`}]
          section = [{title: "Sᴜʙsᴄɪʙᴇ Yᴛ Pᴇᴘᴇ Sɪʀ Fᴏʀ Mᴏʀᴇ Uᴘᴅᴀᴛᴇs", rows: rows}]
          button = {buttonText: 'SELECT', description: `*Gʀᴏᴜᴘ [ᴏᴘᴇɴ/ᴄʟᴏsᴇ]*\n*Dɪsᴀᴘᴘᴇᴀʀ Mᴇsᴀᴀɢᴇ [ᴏɴ/ᴏғғ]*`,
          sections: section, listType: 1}
          bosco.sendMessage(from, button, MessageType.listMessage, {quoted: ftroli})
                    break
         case 'online':
            if (!isOwner && !mek.key.fromMe) return
                                offline = false
                                fgclink('*BOT ONLINE*')
                                break
          case 'offline':
            if (!mek.key.fromMe) return
            offline = true
            waktu = Date.now()
            anuu = args.join(' ') ? args.join(' ') : '-'
```

```
      alasan = anuu
      fgclink('*BOT OFFLINE*')
      break
     case 'tobc':
                    bosco.updatePresence(from, Presence.composing)
                    if (!isOwner) return reply(mess.only.owner)
                    anu = await bosco.chats.all()
                    if (isMedia && !mek.message.videoMessage || isQuotedAudio) {
                    const encmedia = isQuotedAudio ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                    buff = await bosco.downloadMediaMessage(encmedia)
                    for (let _ of anu) {
                    bosco.sendMessage(_.jid, buff, audio, { quoted: { key: { fromMe:
false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "16505434800@s.whatsapp.net" } : {})},
message: { orderMessage: { itemCount: 1000, status: 200, thumbnail: fs.readFileSync('./hemme.jpg'),
surface: 200, message: `${body.slice(5)}`, orderTitle: `hm`, sellerJid: '0@s.whatsapp.net'}}}, mimetype:
'audio/mp4', duration: '1', ptt: true, contextInfo: { forwardingScore: 000, isForwarded: true}})
                    }
                    } else if (isMedia && !mek.message.videoMessage || isQuotedSticker)
{
                    const encmedia = isQuotedSticker ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                    buff = await bosco.downloadMediaMessage(encmedia)
                    for (let _ of anu) {
                    bosco.sendMessage(_.jid, buff, sticker, { quoted: { key: { fromMe:
false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {})}, message: {
orderMessage: { itemCount: 2021, status: 200, thumbnail: fs.readFileSync('./ds.jpg'), surface: 200,
message: `${body.slice(5)}`, orderTitle: `ʙʀᴏᴀᴅᴄᴀsᴛ`, sellerJid: '0@s.whatsapp.net'}}}, contextInfo: {
forwardingScore: 508, isForwarded: true}})
                    }
                    } else if (isMedia && !mek.message.videoMessage || isQuotedVideo) {
                    const encmedia = isQuotedVideo ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                    buff = await bosco.downloadMediaMessage(encmedia)
                    for (let _ of anu) {
                    bosco.sendMessage(_.jid, buff, video, { quoted: { key: { fromMe:
false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {})}, message: {
orderMessage: { itemCount: 2021, status: 200, thumbnail: fs.readFileSync('./ds.jpg'), surface: 200,
message: `${body.slice(5)}`, orderTitle: `ʙʀᴏᴀᴅᴄᴀsᴛ`, sellerJid: '0@s.whatsapp.net'}}}, contextInfo: {
forwardingScore: 508, isForwarded: true}})
                    }
                    } else if (isMedia && !mek.message.videoMessage || isQuotedGif) {
                    const encmedia = isQuotedGif ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                    buff = await bosco.downloadMediaMessage(encmedia)
                    for (let _ of anu) {
                    bosco.sendMessage(_.jid, buff, gif, { quoted: { key: { fromMe:
false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {})}, message: {
orderMessage: { itemCount: 1000, status: 200, thumbnail: fs.readFileSync('./ds.jpg'), surface: 200,
message: `𝑬𝑽𝑬𝑹𝒀𝑻𝑯𝑰𝑵𝑮\n𝑾𝑰𝑳𝑳 𝑩𝑬\n😎𝑶𝑲😎`, orderTitle: `ʙʀᴏᴀᴅᴄᴀsᴛ`, sellerJid: '0@s.whatsapp.net'}}},
contextInfo: { forwardingScore: 22, isForwarded: true}})
                    }
                    } else {
          reply('*Reply To Sticker/Audio/Video*')
                  }
                    break
       case 'restart':
if (!isOwner && !mek.key.fromMe) return reply(mess.only.owner)
reply(`*Restarting*`)
exec(`cd &&  node main`)
sleep(4000)
reply('*Success*')
break
   case "setbgmpic":
    if (
     ((isMedia && !mek.message.videoMessage) ||
      isQuotedImage ||
      isQuotedSticker) &&
     args.length == 0
    ) {
     boij =
      isQuotedImage || isQuotedSticker
       ? JSON.parse(JSON.stringify(mek).replace("quotedM", "m")).message
         .extendedTextMessage.contextInfo
       : mek;
     delb = await bosco.downloadMediaMessage(boij);
     fs.writeFileSync(`./hemme.jpg`, delb);
     reply("Success");
    } else {
     reply(`Send a picture with a caption ${prefix}sethumb`);
    }
    break
   case "setthumb":
        if (
          ((isMedia && !mek.message.videoMessage) ||
            isQuotedImage ||
            isQuotedSticker) &&
          args.length == 0
        ) {
          boij =
            isQuotedImage || isQuotedSticker
              ? JSON.parse(JSON.stringify(mek).replace("quotedM", "m")).message
                  .extendedTextMessage.contextInfo
              : mek;
          delb = await bosco.downloadMediaMessage(boij);
          fs.writeFileSync(`./ds.jpg`, delb);
          reply("Success");
        } else {
          reply(`Send a picture with a caption ${prefix}sethumb`);
        }
        break
        case 'exif':
             if (args.length == 0) return reply(`Ex:- ${prefix}exif Denis Ser`)
             if (!isOwner) return  reply(mess.only.owner)
             if (!q) return reply(mess.wrongFormat)
             if (!arg.split('|')) return reply(`Use ${prefix}exif nama|author`)
             exif.create(arg.split('|')[0], arg.split('|')[1])
             reply('*Succes*')
             break
          case 'axif':
                    if (!isOwner && !mek.key.fromMe) return reply(mess.only.ownerB)
                                        const exifff = `${args.join(' ')}`
                                        const namaPack = exifff.split('|')[0]
                                        const authorPack = exifff.split('|')[1]
                                        exif.create(namaPack, authorPack)
                                        await reply('Done gan')
                                break
     case 'searchmsg':  //by ANU TEAM
             if (args.length < 1) return reply(`*What Message Are You Looking For?*\n*Example* : ${prefix +
command} halo|10`)
             teks = arg
             if (teks.includes("|")) {
             try {
             var ve = teks.split("|")[0]
             var za = teks.split("|")[1]
             sampai = `${za}`
             if (isNaN(sampai)) return reply('*Must be a Number!*')
             batas = parseInt(sampai) + 1
             if (batas > 30) return reply('*Max 30!*')
             reply(mess.wait)
             cok = await bosco.searchMessages(`${ve}`, from, batas,1)
             if (cok.messages.length < 2) return reply('*Message Not Found*')
             if (cok.messages.length < parseInt(batas)) reply(`*Found Only* ${cok.messages.length - 1}
*Message*`)
             for (i=1;i < cok.messages.length;i++) {
             if (cok.messages[i].message) {
             bosco.sendMessage(from, `*Found..!*`, text, {sendEphemeral: true, quoted: cok.messages[i]})
}
}

             } catch (e) {
             return reply(String(e))
}
             } else {
             reply(`*The format is wrong tod, this is an example of the correct format* : ${prefix +
command} halo|10`)
}
             break
     case 'seenby':
             if(!isGroup) return reply(mess.only.group)
             try {
             infom = await bosco.messageInfo(from, mek.message.extendedTextMessage.contextInfo.stanzaId)
             tagg = []
             teks = `*• Read by :*\n\n`
             for(let i of infom.reads){
             teks += '@' + i.jid.split('@')[0] + '\n'
             teks += `> ` + moment(`${i.t}` * 1000).tz('Asia/Kolkata').format('DD/MM/YYYY hh:mm:ss') +
'\n\n'
             tagg.push(i.jid)
}
             teks += `*• Delivered to :*\n\n`
             for(let i of infom.deliveries){
             teks += '@' + i.jid.split('@')[0] + '\n'
             teks += `> ` + moment(`${i.t}` * 1000).tz('Asia/Kolkata').format('DD/MM/YYYY hh:mm:ss') +
'\n\n'
             tagg.push(i.jid)
}
```

```
             mentions(teks, tagg, true)
             } catch (e) {
             console.log(color(e))
             reply('*Reply chat bot!*')
}
             break
      case 'leave':
              if (!isGroup) return reply(mess.only.group)
              setTimeout( () => {
              bosco.groupLeave(from)
              }, 2000)
              setTimeout( () => {
              reply('*Byee all..:(* 🚶')
              }, 0)
              break
       case 'online':
       case 'listonline':
       case 'here':
             if (!isGroup) return reply(`*Only group*`)
             try {
             let ido = args && /\d+\-\d+@g.us/.test(args[0]) ? args[0] : from
             let online = [...Object.keys(bosco.chats.get(ido).presences), bosco.user.jid]
             bosco.sendMessage(from, '*List Online:*\n' + online.map(v => '- @' + v.replace(/@.+/,
'')).join `\n`, text, { quoted: mek, contextInfo: { mentionedJid: online }})
             } catch (e) {
             reply(`${e}`)
}
             break
      case 'join':
             if (args.length == 0) return reply(`Ex:- ${prefix}join https://chat.whatsapp.com`)
             if (!q) return reply('*The link?*')
             if (!isOwner) return reply(mess.only.owner)
             if (!isUrl(args[0]) && !args[0].includes('https://chat.whatsapp.com/')) return reply('*The
link is invalid Tod*')
             link = args[0].replace('https://chat.whatsapp.com/','')
             fak = bosco.query({ json: ['action', 'invite', link],
             expect200: true })
             reply('*Successfully Entered Group*')
             break
      case 'readall':
                                   if (!mek.key.fromMe) return reply('```OWNER ONLY```')
                                   var chats = await bosco.chats.all()
                    chats.map( async ({ jid }) => {
                          await bosco.chatRead(jid)
                    })
                                   rdl = `*Successfully read ${chats.length} Chat !*`
                                   await bosco.sendMessage(from, rdl, MessageType.text, {quoted:
ftroli})

                                   console.log(chats.length)
                                   break
      case 'imgtourl':
       case 'img2url':
              reply(mess.wait)
              var imgbb = require('imgbb-uploader')
              var encmediahe  = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo : mek
              var mediahe = await  bosco.downloadAndSaveMediaMessage(encmediahe)
              imgbb('39d895963468b814fad0514bd28787e2', mediahe)
             .then(data => {
              var caps = `*IMAGE TO URL*\n\n*~>  ID :* ${data.id}\n*~>  MimeType :*
${data.image.mime}\n*~>  Extension :* ${data.image.extension}\n*~>  URL :* ${data.display_url}`
              ibb = fs.readFileSync(mediahe)
              bosco.sendMessage(from, ibb, image, { quoted: mek, caption: caps})
})
             .catch(err => {
              throw err
})
              break
      case 'term':
             if (!isOwner) return
             if (args.length == 0) return reply(`Ex:- ${prefix}term sendKontak`)
             if (!q) return
             exec(q, (err, stdout) => {
             if (err) return reply(`${err}`)
             if (stdout) {
             reply(stdout)
}
})
             break
   case 'youtube':
      case 'ytmp3':
      case 'yt':
      case 'video':
```

```
             if (args.length < 1) return reply('*Where is the link?*')
             if(!isUrl(args[0]) && !args[0].includes('youtu')) return reply(mess.error.Iv)
             teks = args.join(' ')
             res = await y2mateA(teks).catch(e => {
             reply('*Error Failed To Enter Y2mate Web*')
})
             result = `
*Tɪᴛʟᴇ :* ${res[0].judul}
*Tʏᴘᴇ :* mp3/mp4
*Sɪᴢᴇ :* ${res[0].size}`
              buttons = [{buttonId: `${prefix}buttons2 ${q}`,buttonText:{displayText: 'ᴠɪᴅᴇᴏ'},type:1},
{buttonId:`${prefix}buttons1 ${q}`,buttonText:{displayText:'ᴀᴜᴅɪᴏ'},type:1}]
              fs.writeFileSync(`./ytmp.jpeg`, await getBuffer(res[0].thumb))
              yt1 = await bosco.prepareMessage(from, fs.readFileSync(`./ytmp.jpeg`), location, {thumbnail:
fs.readFileSync(`./ytmp.jpeg`),})
              yt2 = yt1.message["ephemeralMessage"] ? yt1.message.ephemeralMessage : yt1
              buttonsMessage = {footerText:`${result}`,
              contentText:` `,buttons,headerType:6, locationMessage: yt2.message.locationMessage}
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{quoted: ftroli})
              bosco.relayWAMessage(prep)
              fs.unlinkSync(`./ytmp.jpeg`)
              break
      case 'rall':
                                const readallid = await bosco.chats.all()
                        bosco.setMaxListeners(25)
                                for (let xyz of readallid) {
                                        await bosco.chatRead(xyz.jid)
                                }
                      bosco.sendMessage(from, `Sukses!`, text, { quoted: { key: { fromMe: false,
participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {}) }, message: {
"imageMessage": { "mimetype": "image/jpeg", "caption": "Berhasil membaca semua chat!", 'jpegThumbnail':
fs.readFileSync('./ds.jpg')}}}})
              break
      case 'shutdown':
             if (!isOwner) return reply(mess.owner.only)
             reply(`Bye...`)
             await sleep(3000)
             process.exit()
             break
      case 'leaveall':
             if (!isOwner) return
             let totalgroup = bosco.chats.array.filter(u => u.jid.endsWith('@g.us')).map(u => u.jid)
             for (let id of totalgroup) {
             sendMess(id, 'Byee', null)
             await sleep(3000)
             bosco.groupLeave(id)
}
             break
       case 'culik':
              if (!isOwner) return
              if (args.length < 1) return reply('*Enter the group id lol*')
              let pantek = []
              for (let i of groupMembers) {
              pantek.push(i.jid)
}
              bosco.groupAdd(args[0], pantek)
              break

      case 'hidetag':
             try {
             quotedText = mek.message.extendedTextMessage.contextInfo.quotedMessage.conversation
             hideTag(from, `${quotedText}`)
             } catch {
             hideTag(from, `${q}`)
}
             break
       case 'denis':
              if (!q) return
              qq = q.toUpperCase()
              awikwok = `${qq} ${qq} ${qq} ❤️ ❤️ ❤️ WANGY WANGY WANGY WANGY HU HA HU HA HU HA, Pepe Poli
Ahn ${qq} Pepe Killadi Ahnu ${qq} AAAAAAAAH ~ Rambutnya.... aaah rambutnya juga pengen aku elus-elus ~~
AAAAAH ${qq} keluar pertama kali di anime juga manis ❤️ ❤️ ❤️ banget AAAAAAAAH ${qq} AAAAA
LUCCUUUUUUUUUUUUUUU............ ${qq} AAAAAAAAAAAAAAAAAAAAGH ❤️ ❤️ ❤️apa ? ${qq} itu gak nyata ? Cuma HALU
katamu ? nggak, ngak ngak ngak NGAAAAAAAAK GUA GAK PERCAYA ITU DIA NYATA NGAAAAAAAAAAAAAAAAAK PEDULI
BANGSAAAAAT !! GUA GAK PEDULI SAMA KENYATAAN POKOKNYA GAK PEDULI. ❤️ ❤️ ❤️ ${qq} gw ... ${qq} di laptop
ngeliatin gw, ${qq} .. kamu percaya sama aku ? aaaaaaaaaaah syukur ${q} aku gak mau merelakan ${qq} aaaaaah
❤️ ❤️ ❤️ YEAAAAAAAAAAAAH GUA MASIH PUNYA ${qq} SENDIRI PUN NGGAK SAMA AAAAAAAAAAAAAAH`
              reply(awikwok)
              break
        case 'jadibot':
              if (!isOwner) return
              jadibot(reply,bosco,from)
              break
       case 'stopjadibot':
             stopjadibot(reply)
             break
      case 'listbot':
      case 'listjadibot':
          text = '* 「 LIST JADIBOT 」*\n\n'
             for(let i of listjadibot) {
             text += `*Number* : ${i.jid.split('@')[0]}
*Name* : ${i.name}
*Device* : ${i.phone.device_manufacturer}
*Model* : ${i.phone.device_model}\n\n`
}
            reply(text)
            break

//------------------< Editing/Audio/Video >-------------------

                case 'tr':
                case 'translate':
                    if (args.length == 0) return reply(`Example: ${prefix + command} en apa`)
                    kode_negara = args[0]
                    args.shift()
                    teks = args.join(" ")
                    translate(`${teks}`,{to:`${kode_negara}`}).then( res => {
                    ini_txt = `*Translate*
                     Lng : ${res.from.language.iso}
                     result : ${res.text}`
                     reply(ini_txt)
                     })
                     break
               case 'secvn':
                    reply(mess.wait)
                                        encmedia = 
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        cokmatane = Number(args[0])
                                        hah = fs.readFileSync(media)
                                        bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4', 
duration: cokmatane, ptt: true, quoted:mek})
                                        fs.unlinkSync(media)
                                    break
                                case 'secvideo':
                                        encmedia = 
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        cokmatane = Number(args[0])
                                        hah = fs.readFileSync(media)
                                        bosco.sendMessage(from, hah, video, {mimetype: 'video/mp4', 
duration: cokmatane, quoted: mek})
                                        fs.unlinkSync(media)
                                    break
               case 'voice':
                                        encmedia = 
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} ${ran}`, (err) => {
                                        fs.unlinkSync(media)
                                        if (err) return reply('*_FAILD TO CONVERT PTT_*')
                                        topt = fs.readFileSync(ran)
                                        bosco.sendMessage(from, topt, audio, { mimetype: 'audio/mp4', ptt: 
true, quoted: mek })
                                        })
                                        break
               case 'take':
              case 'colong':
                    if (!isQuotedSticker) return reply('*Just a sticker*')
                    encmedia = 
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                            media = await bosco.downloadAndSaveMediaMessage(encmedia)
                    anu = args.join(' ').split('|')
                    satu = anu[0] !== '' ? anu[0] : `Denis`
                    dua = typeof anu[1] !== 'undefined' ? anu[1] : `BOT`
                    require('./lib/fetch.js').createExif(satu, dua)
                                require('./lib/fetch.js').modStick(media, bosco, mek, from)
                                break
             case 'fastvid':
                            if (!isQuotedVideo) return fakeitem('Reply videonya!')
                            fakegroup(mess.wait)
                            encmedia = JSON.parse(JSON.stringify(mek).replace('quotedM', 
'm')).message.extendedTextMessage.contextInfo
                            media = await bosco.downloadAndSaveMediaMessage(encmedia)
                            ran = getRandom('.mp4')
                            exec(`ffmpeg -i ${media} -filter_complex "[0:v]setpts=0.5*PTS[v];
[0:a]atempo=2[a]" -map "[v]" -map "[a]" ${ran}`, (err) => {
                            fs.unlinkSync(media)
                            if (err) return fakegroup(`Err: ${err}`)
                            buffer453 = fs.readFileSync(ran)
                            bosco.sendMessage(from, buffer453, video, { mimetype: 'video/mp4', quoted: mek
})
                            fs.unlinkSync(ran)
                            })
                            break
                    case 'slowvid':
                            if (!isQuotedVideo) return fakeitem('Reply videonya!')
                            fakegroup(mess.wait)
                            encmedia = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
                            media = await bosco.downloadAndSaveMediaMessage(encmedia)
                            ran = getRandom('.mp4')
                            exec(`ffmpeg -i ${media} -filter_complex "[0:v]setpts=2*PTS[v];
[0:a]atempo=0.5[a]" -map "[v]" -map "[a]" ${ran}`, (err) => {
                            fs.unlinkSync(media)
                            if (err) return fakegroup(`Err: ${err}`)
                            buffer453 = fs.readFileSync(ran)
                            bosco.sendMessage(from, buffer453, video, { mimetype: 'video/mp4', quoted: mek
})
                            fs.unlinkSync(ran)
                            })
                            break
                    case 'reversevid':
                            if (!isQuotedVideo) return fakeitem('```Reply videonya!```')
                            encmedia = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
                            media = await bosco.downloadAndSaveMediaMessage(encmedia)
                            ran = getRandom('.mp4')
                            exec(`ffmpeg -i ${media} -vf reverse -af areverse ${ran}`, (err) => {
                            fs.unlinkSync(media)
                            if (err) return fakegroup(`Err: ${err}`)
                            buffer453 = fs.readFileSync(ran)
                            bosco.sendMessage(from, buffer453, video, { mimetype: 'video/mp4', quoted: mek
})
                            fs.unlinkSync(ran)
                            })
                            break
             case 'nightcore':
                         if (!isQuotedAudio) return reply('Reply audio nya om')
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -filter:a atempo=1.06,asetrate=44100*1.25
${ran}`, (err, stderr, stdout) => {
                                                fs.unlinkSync(media)
                                                if (err) return reply('Error!')
                                                hah = fs.readFileSync(ran)
                                                bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4',
ptt:true, quoted: ftoko,duration:11})
                                                fs.unlinkSync(ran)
                                           })
                                       break

             case 'imut':
             case 'baby':
                   if (!isQuotedAudio) return fgclink('reply to audio')
                                    encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                    media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -af atempo=3/4,asetrate=44508*4/3 ${ran}`,
(err, stderr, stdout) => {
                                    fs.unlinkSync(media)
                                        if (err) return reply('Error!')
                                        hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4', ptt:
true, quoted: mek})
                                        fs.unlinkSync(ran)
                                })
                                     break
           case 'cm':
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp4')
                                        exec(`ffmpeg -i ${media} "origin(rgb24).png" -c:v libx264 -preset
placebo -qp 0 -x264-params "keyint=15:no-deblock=1" -pix_fmt yuv444p10le -sws_flags
```

```
spline+accurate_rnd+full_chroma_int -vf "colormatrix=bt470bg:bt709" -color_range 1 -colorspace 1 -
color_primaries 1 -color_trc 1 "colormatrix_yuv444p10le.avi" ${ran}`, (err, stderr, stdout) => {
                                                fs.unlinkSync(media)
                                                if (err) return reply('Error!')
                                                hah = fs.readFileSync(ran)
                                                bosco.sendMessage(from, hah, video, { mimetype:
'video/mp4', quoted: ftoko })
                                        })
                                        break
           case 'fast':
                     if (!isQuotedAudio) return fgclink('reply to audio')
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -filter:a "atempo=1.3,asetrate=43000"
${ran}`, (err, stderr, stdout) => {
                                                fs.unlinkSync(media)
                                                if (err) return reply('Error!')
                                                hah = fs.readFileSync(ran)
                                                bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4',
ptt:true, quoted: ftext})
                                                fs.unlinkSync(ran)
                                        })
                                        break
             case 'gemes':
                    if (!isQuotedAudio) return fgclink('reply to audio')
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -filter:a "atempo=1.0,asetrate=50000"
${ran}`, (err, stderr, stdout) => {
                                                fs.unlinkSync(media)
                                                if (err) return reply('Error!')
                                                hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4',
ptt:true, quoted: fgc})
                                                fs.unlinkSync(ran)
                                        })
                                        break
            case 'balik':
            case 'reverse':
                  if (!isQuotedAudio) return fgclink('reply to audio')
                      encmediau =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                      mediau = await bosco.downloadAndSaveMediaMessage(encmediau)
                      ran = getRandom('.mp3')
                      exec(`ffmpeg -i ${mediau} -filter_complex "areverse" ${ran}`, (err, stderr, stdout)
=> {
                  fs.unlinkSync(mediau)
                  if (err) return reply('Error!')
                  hah = fs.readFileSync(ran)
                  bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4', ptt: true, duration:
359996400, quoted:mek})
                  fs.unlinkSync(ran)
                      })
                  break
            case 'bass':
            case 'volume':
                    if (!isQuotedAudio) return fgclink('reply to audio')
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -af
equalizer=f=65:width_type=o:width=3:g=20 ${ran}`, (err, stderr, stdout) => {
                                        fs.unlinkSync(media)
                                        if (err) return reply('Error!')
                                        hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, { mimetype: 'audio/mp4', ptt:
true, quoted: mek })
                                        fs.unlinkSync(ran)
                                      })
                                    break
             case 'slow':
                    if (!isQuotedAudio) return fgclink('reply to audio')
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -filter:a "atempo=0.7,asetrate=43120"
${ran}`, (err, stderr, stdout) => {
```

```
                                        fs.unlinkSync(media)
                                        if (err) return reply('Error!')
                                        hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, { mimetype: 'audio/mp4', ptt:
true, quoted: mek })
                                        fs.unlinkSync(ran)
                                         })
                                    break
                case 'squirrel':
                                         encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                         media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                         ran = getRandom('.mp3')
                                         exec(`ffmpeg -i ${media} -filter:a "atempo=0.5,asetrate=65100"
${ran}`, (err, stderr, stdout) => {
                                        fs.unlinkSync(media)
                                        if (err) return reply('Error!')
                                        hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, { mimetype: 'audio/mp4', ptt:
true, quoted: mek })
                                        fs.unlinkSync(ran)
                                         })
                                    break
                                case 'blub':
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -filter:a "atempo=0.9,asetrate=95100"
${ran}`, (err, stderr, stdout) => {
                                        fs.unlinkSync(media)
                                        if (err) return reply('Error!')
                                        hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, { mimetype: 'audio/mp4',
duration: 359996400, ptt: true, quoted: mek })
                                        fs.unlinkSync(ran)
                                         })
                                    break
                                case 'fat':
                         if (!isQuotedAudio) return reply('*_ʀᴇᴘʟʏ ᴛᴏ ᴀᴜᴅɪᴏ_*')
                                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        ran = getRandom('.mp3')
                                        exec(`ffmpeg -i ${media} -filter:a "atempo=1.1,asetrate=35101"
${ran}`, (err, stderr, stdout) => {
                                        fs.unlinkSync(media)
                                        if (err) return reply('*ʀᴇᴛʀʏ!*')
                                        hah = fs.readFileSync(ran)
                                        bosco.sendMessage(from, hah, audio, { mimetype: 'audio/mp4', ptt:
true, quoted: mek })
                                        fs.unlinkSync(ran)
                                     })
                                     break
      case 'tomp4':
                                        if (!isQuotedSticker) return reply('Reply stiker nya')
                                        reply(mess.wait)
            if ((isMedia && !mek.message.videoMessage || isQuotedSticker) && args.length == 0) {
            ger = isQuotedSticker ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            owgi = await bosco.downloadAndSaveMediaMessage(ger)
            webp2mp4File(owgi).then(res=>{
            sendMediaURL(from,res.result)
            })
            }else {
            reply('Reply Stickernya!')
            }
            fs.unlinkSync(owgi)
            break
        case 'tomp3':
        case 'mp3':
                                        bosco.updatePresence(from, Presence.composing)
                                        encmediad = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
                                        mediad = await bosco.downloadAndSaveMediaMessage(encmediad)
                                        ran = getRandom('.mp4')
                                        exec(`ffmpeg -i ${mediad} ${ran}`, (err) => {
                                                fs.unlinkSync(mediad)
                                                if (err) return reply(mess.error.api)
                                                mhee = fs.readFileSync(ran)
                                                bosco.sendMessage(from, mhee, audio, { mimetype:
'audio/mp4', duration: 4, quoted: mek })
                                                fs.unlinkSync(ran)
                                        })
                                        break
      case 'kick':
if (!isOwner && !isGroupAdmins) return reply('*Admin Group Only*')
if (!isBotGroupAdmins) return reply('*Bot not admin!*')
if (!isGroup) return
if (mek.message.extendedTextMessage === null || mek.message.extendedTextMessage === undefined) return;
if (mek.message.extendedTextMessage.contextInfo.participant === undefined) {
entah = mek.message.extendedTextMessage.contextInfo.mentionedJid
if (entah.length > 1) {
var mems_ids = []
for (let ids of entah) {
mems_ids.push(ids)
}
bosco.groupRemove(from, mems_ids)
} else {
bosco.groupRemove(from, [entah[0]])
}
} else {
entah = mek.message.extendedTextMessage.contextInfo.participant
bosco.groupRemove(from, [entah])
}
break
case 'add':
      if (!isOwner && !isGroupAdmins) return reply('*Admin Group Only*')
if (!isBotGroupAdmins) return reply('*Bot not admin!*')
if (!isGroup) return
if (mek.message.extendedTextMessage === null || mek.message.extendedTextMessage === undefined) return;
if (mek.message.extendedTextMessage.contextInfo.participant === undefined) {
entah = mek.message.extendedTextMessage.contextInfo.mentionedJid
if (entah.length > 1) {
var memu_ido = []
for (let ids of entah) {
mems_ids.push(ido)
}
bosco.groupAdd(from, memu_ido)
} else {
bosco.groupAdd(from, [entah[0]])
}
} else {
entah = mek.message.extendedTextMessage.contextInfo.participant
bosco.groupAdd(from, [entah])
}
break
case 'promote':
if (!isGroupAdmins) return reply('*Admin Group Only*')
if (!isBotGroupAdmins) return reply('*Bot not admin!*')
if (!isGroup) return
if (mek.message.extendedTextMessage === null || mek.message.extendedTextMessage === undefined) return;
if (mek.message.extendedTextMessage.contextInfo.participant === undefined) {
entah = mek.message.extendedTextMessage.contextInfo.mentionedJid
if (entah.length > 1) {
var memi_idi = []
for (let ids of entah) {
memi_idi.push(idi)
}
bosco.groupMakeAdmin(from, memi_idi)
} else {
bosco.groupMakeAdmin(from, [entah[0]])
}
} else {
entah = mek.message.extendedTextMessage.contextInfo.participant
bosco.groupMakeAdmin(from, [entah])
}
break
case 'demote':
      if (!isOwner && !isGroupAdmins) return reply('*Admin Group Only*')
if (!isBotGroupAdmins) return reply('*Bot not admin!*')
if (!isGroup) return
if (mek.message.extendedTextMessage === null || mek.message.extendedTextMessage === undefined) return;
if (mek.message.extendedTextMessage.contextInfo.participant === undefined) {
entah = mek.message.extendedTextMessage.contextInfo.mentionedJid
if (entah.length > 1) {
var memu_ido = []
for (let idk of entah) {
memk_idk.push(idk)
}
bosco.groupDemoteAdmin(from, memk_idk)
} else {
bosco.groupDemoteAdmin(from, [entah[0]])
}
} else {
entah = mek.message.extendedTextMessage.contextInfo.participant
```

```
bosco.groupDemoteAdmin(from, [entah])
}
break

      case 'bc':
                                        bosco.updatePresence(from, Presence.composing)
                                        if (!isOwner && !mek.key.fromMe) return reply(mess.only.ownerB)
                                        if (args.length < 1) return reply('*Where is Text?*')
                                        anu = await bosco.chats.all()
                                        if (isMedia && !mek.message.videoMessage || isQuotedImage) {
                                             encmedia = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                                                buff = await bosco.downloadMediaMessage(encmedia)
                                                for (let _ of anu) {
                                                        bosco.sendMessage(_.jid, buff, image, { caption:
`${body.slice(4)}`})
                                                }
                                                reply(`*Broadcast success* ${body.slice(4)}`)
                                                } else if (isMedia && !mek.message.videoMessage ||
isQuotedVideo) {
                                                 encmedia = isQuotedVideo ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                                                buff = await bosco.downloadMediaMessage(encmedia)
                                                for (let _ of anu) {
                                                        bosco.sendMessage(_.jid, buff, video, { caption:
`${body.slice(4)}`})
                                                }
                                                reply(`*Broadcast success* ${body.slice(4)}`)
                                                } else if (isMedia && !mek.message.videoMessage ||
isQuotedVideo) {
                                                 encmedia = isQuotedVideo ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                                                buff = await bosco.downloadMediaMessage(encmedia)
                                                for (let _ of anu) {
                                                        bosco.sendMessage(_.jid, buff, video, { mimetype:
Mimetype.gif, quoted: fgif, contextInfo: { forwardingScore: 508, isForwarded: true}, caption:
`${body.slice(4)}` })
                                                }
                                                reply(`*Broadcast success* ${body.slice(4)}`)
                                        } else {
                                                for (let _ of anu) {
                                                        sendMess(_.jid, `${body.slice(4)}`)
                                                }
                                                reply(`*Broadcast success*:\n${body.slice(4)}`)
                                        }
                                        break
       case 'contact':
                                if (!isGroup) return reply(mess.group)
                                        argzu = arg.split('|')
                                if (!argzu) return reply(`*Use ${prefix}contact @tag|name*`)
                                if (mek.message.extendedTextMessage != undefined){
                    mentioned = mek.message.extendedTextMessage.contextInfo.mentionedJid
                                        sendKontak(from, mentioned[0].split('@')[0], argzu[1])
                                } else {
                                        sendKontak(from, argzu[0], argzu[1])
                                }
                                break
       case 'sharelock':
kntl = `${args.join(' ')}`
nama = kntl.split("|")[0];
impostor = kntl.split("|")[1];
bosco.sendMessage(from, {
name: nama,
address: impostor,
jpegThumbnail: denis}, MessageType.liveLocation, {quoted:floc2})
break
       case 'getdeskgc':
                                if (!isGroup) return reply(mess.group)
                                anu = from
                           metadete = await bosco.groupMetadata(anu)
                                bosco.sendMessage(from, metadete.desc, MessageType.text, {quoted:fgif})
                                  break
                                        case 'getbio':
                          var yy = mek.message.extendedTextMessage.contextInfo.participant
                       var p = await bosco.getStatus(`${yy}`, MessageType.text, {quoted: mek})
                        reply(p.status)
                         if (p.status == 401) {
                          reply(mess.error.api)
                          }
                          break
           case 'chat':
                        if (args[0].startsWith('08')) return reply('*Prefix number with 91*')
            if (args[0].startsWith('+91')) return reply('*Prefix number with 91*')
```

```
                        if (args.length < 1) return reply(`Penggunaan ${prefix}chat 91xxxx|teks`)
            var pc = body.slice(6)
            var nomor = pc.split("|")[0];
            var org = pc.split("|")[1];
            bosco.sendMessage(nomor+'@s.whatsapp.net', org, MessageType.text)
            reply(`*Success in sending chat* :\n${org},@${nomor}`)
            break
                    case 'getname':
                      var ambl = mek.message.extendedTextMessage.contextInfo.participant
                      const sname = bosco.contacts[ambl] != undefined ? bosco.contacts[ambl].notify =
undefined ? PhoneNumber('+' + ambl.replace('@s.whatsapp.net', '')).getNumber('international') :
bosco.contacts[ambl].notify || bosco.contacts[ambl].vname : PhoneNumber('+' +
ambl.replace('@s.whatsapp.net', '')).getNumber('international')
                       reply(sname)
                        break
      case 'toimg':
      case 'photo':
              if (!isQuotedSticker) return reply('*Reply To Sticker*')
              encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
              media = await bosco.downloadAndSaveMediaMessage(encmedia)
              ran = getRandom('.png')
              exec(`ffmpeg -i ${media} ${ran}`, (err) => {
              fs.unlinkSync(media)
              if (err) return reply('*Failed, when converting sticker to image*')
              buffer = fs.readFileSync(ran)
              bosco.sendMessage(from, buffer, image, {quoted: mek, thumbnail:denis, caption: 'By Bosco'})
              fs.unlinkSync(ran)
})
              break
//-------------------< Public/Self >-------------------

        case 'public':
                if (!mek.key.fromMe) return
                if (banChats === false) return
                banChats = false
              fgclink(`*[ PUBLIC - MODE ]*`)
                break
        case 'self':
                if (!mek.key.fromMe) return
                if (banChats === true) return
                uptime = process.uptime()
                banChats = true
              fgclink(`*[ SELF - MODE ]*`)
                break

//-------------------< Downloader/And Search Media >-------------------

       case 'igdl':
       case 'instagram':
              if (!q) return reply('*Which Links?*')
              if (!q.includes('instagram')) return reply(mess.error.Iv)
              reply(mess.wait)
              anu = await igDownloader(`${q}`)
             .then((data) => { sendMediaURL(from, data.result.link, data.result.desc, mek) })
             .catch((err) => { reply(String(err)) })
              break
       case 'igstory':
              if(!q) return reply('*Username?*')
              hx.igstory(q)
             .then(async result => {
              for(let i of result.medias){
              if(i.url.includes('mp4')){
              let link = await getBuffer(i.url)
              bosco.sendMessage(from,link,video,{quoted: mek,caption: `Type : ${i.type}`})
              } else {
              let link = await getBuffer(i.url)
              bosco.sendMessage(from,link,image,{quoted: mek,caption: `Type : ${i.type}`})
              }
              }
              });
              break
      case 'hehe':
      ratee =
["100","101","102","103","104","105","106","107","108","109","110","111","112","113","114","115","116","117
","118","119","120"]
      const tee = ratee[Math.floor(Math.random() * ratee.length)]
      hemmo = fs.readFileSync(`mp3/100.mp3`)
      bosco.sendMessage(from, hemmo, audio, { mimetype: 'audio/mp4', ptt: true, quoted: mek})
      break
      case 'readmore':
      case 'more':
                                const more = String.fromCharCode(8206)
```

```
                              const readmore = more.repeat(4001)
                                  if (!c.includes('|')) return  reply(mess.error.api)
                  const text1 = c.substring(0, c.indexOf('|') - 0)
                  const text2 = c.substring(c.lastIndexOf('|') + 1)
                  reply( text1 + readmore + text2)
                  break
        case 'lolkey':
        case 'cekapikey':
               if (args.length < 1) return reply(`*Type ${prefix}lolkey [Apikey]*`)
               anu = await fetchJson(`https://lolhuman.herokuapp.com/api/checkapikey?apikey=${q}`)
               teks = `*YOUR APIKEY*\n\n➸ Ussername= ${anu.result.username}\n➸ Request=
${anu.result.requests}\n➸ Today= ${anu.result.today}\n➸ Account Type= ${anu.result.account_type}\n➸
Expired= ${anu.result.expired}\n➸ API = https://lolhuman.herokuapp.com`
               bosco.sendMessage(from, teks, text, {quoted: mek})
               break
       case 'pinterest':
         case 'pin':
              if (args.length < 1) return reply(`${prefix}Denis Ser`)
              data = await fetchJson(`https://lolhuman.herokuapp.com/api/pinterest?
apikey=${lolkey}&query=${q}`)
              buttons = [{buttonId: `${prefix + command} ${q}`,buttonText:{displayText: `➡️Next`},type:1}]
              fs.writeFileSync(`./${sender}.jpeg`, await getBuffer(data.result))
              imageMsg = ( await bosco.prepareMessage(from, fs.readFileSync(`./${sender}.jpeg`),
'imageMessage', {thumbnail: Buffer.alloc(0)})).message.imageMessage
              buttonsMessage = {footerText:'Hello Sis Can i help u..', imageMessage: imageMsg,
              contentText:`*Search Results From : ${q}*`,buttons,headerType:4}
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{})
              bosco.relayWAMessage(prep)
              fs.unlinkSync(`./${sender}.jpeg`)
              break
       case 'yts':
       case 'ytsearch':
              if (!q) return reply(mess.wrongFormat)
              reply(mess.wait)
              try {
              res = await yts(q)
              a = `*Youtube Search 🔎*\n`
for (let i of res.all) {
a += `
📜 Title : ${i.title}
🎞️ Views : ${i.views}
🌐 Upload : ${i.ago}
⏱️ Duration : ${i.timestamp}
🎥 Channel : ${i.author.name}
🖇️ Link : ${i.url}\n`
}
               b = a.trim()
               sendFileFromUrl(res.all[0].image, image, {quoted: mek, thumbnail: Buffer.alloc(0), caption:
b})
               } catch (e) {
               console.log(e)
               reply(`${e}`)
}
               break
        case 'scplay':
       case 'soundcloud':
              if (!q) return reply('Which Links?')
              if (!q.includes('soundcloud')) return reply(mess.error.Iv)
              reply(mess.wait)
              anu = await fetchJson(`https://api.lolhuman.xyz/api/soundcloud?apikey=${lolkey}&url=${q}`)
             .then((data) => { sendMediaURL(from, data.result, data.title, mek) })
             .catch((err) => { reply(String(err)) })
              break
       case 'tourl':
               if ((isMedia && !mek.message.videoMessage || isQuotedImage || isQuotedVideo ) && args.length
== 0) {
               reply(mess.wait)
               boij = isQuotedImage || isQuotedVideo ?
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo : mek
               owgi = await bosco.downloadMediaMessage(boij)
               res = await uploadImages(owgi)
               reply(res)
               } else {
               reply('*_Send to Reply Pictures/Videos_*')
}
               break
       case 'telesticker':
       case 'telestiker':
              if (!q) return reply(`Example: ${prefix + command}
https://t.me/addstickers/LINE_Menhera_chan_ENG`)
              reply(mess.wait)
              ini_url = await fetchJson(`https://api.lolhuman.xyz/api/telestick?
apikey=${lolkey}&url=${args[0]}`)
              ini_sticker = ini_url.result.sticker
              reply('Sending '+ ini_sticker.length +' stickers...')
              for (sticker_ in ini_sticker) {
              ini_buffer = await getBuffer(ini_sticker[sticker_])
              bosco.sendMessage(from, ini_buffer, sticker, {})
}
              break
       case 'ghsearch':
       case 'githubsearch':
       case 'searchgithub':
             if (!q) return reply('*What are you looking for?*')
             res = await fetch('https://api.github.com/search/repositories?q='+q)
             json = await res.json()
             if (res.status !== 200) throw json
             str = json.items.map((repo, index) => {
             return `
${1 + index}. *${repo.full_name}*${repo.fork ? ' (fork)' : ''}
${repo.html_url}
Made on *${formatDate(repo.created_at)}*
Last updated on *${formatDate(repo.updated_at)}*
👁  ${repo.watchers}   🍴  ${repo.forks}   ⭐  ${repo.stargazers_count}
${repo.open_issues} Issue${repo.description ? `
*Description:*\n${repo.description}` : ''}
*Clone:* *$ git clone ${repo.clone_url}*
`.trim()
}).join('\n\n')
            reply(str)
            break
     case 'image':
     case 'gimage':
     case 'googleimage':
              if (args.length < 1) return reply('*What do you want to search?*')
              reply(mess.wait)
              teks = args.join(' ')
              res = await _gis(teks, google)
              function google(error, result){
              if (error){ return reply('*Error An Error Occurs Or No Result Found*')}
              else {
              gugIm = result
              random =  gugIm[Math.floor(Math.random() * gugIm.length)].url
              sendFileFromUrl(random, image, {quoted: mek, thumbnail: Buffer.alloc(0), caption: `*Search
Results From :* ${teks}`})
              }
              }
             break
      case 'tiktokdl':
              if (!q) return reply('The link?')
              if (!q.includes('tiktok')) return reply(mess.error.Iv)
              data = await fetchJson(`https://api.lolhuman.xyz/api/tiktok?apikey=${lolkey}&url=${q}`)
              result = `⚜️ *Nickname*: ${data.result.author.nickname}\n❤️ *Like*:
${data.result.statistic.diggCount}\n💬 *Comment*: ${data.result.statistic.commentCount}\n🔁 *Share*:
${data.result.statistic.shareCount}\n🎞️ *Views*: ${data.result.statistic.playCount}\n📑 *Desc*:
${data.result.title}`
              buttons = [{buttonId: `${prefix}buttons3 ${q}`,buttonText:{displayText: `▶️ Video`},type:1},
{buttonId:`${prefix}buttons4 ${q}`,buttonText:{displayText:'🎵 Audio'},type:1}]
              fs.writeFileSync(`./${sender}.jpeg`, await getBuffer(data.result.thumbnail))
              imageMsg = ( await bosco.prepareMessage(from, fs.readFileSync(`./${sender}.jpeg`),
'imageMessage', {thumbnail: Buffer.alloc(0)})).message.imageMessage
              buttonsMessage = {footerText:'Choose a format below', imageMessage: imageMsg,
              contentText:`${result}`,buttons,headerType:4}
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{quoted: mek})
              bosco.relayWAMessage(prep)
              fs.unlinkSync(`./${sender}.jpeg`)
              break
      case 'ttnowm':
      case 'tiktoknowm':
             if (!q) return reply('The link?')
             if (!q.includes('tiktok')) return reply(mess.error.Iv)
             reply(mess.wait)
             anu = await TiktokDownloader(`${q}`)
            .then((data) => { sendMediaURL(from, data.result.nowatermark) })
            .catch((err) => { reply(String(err)) })
             break
      case 'ttaudio':
      case 'tiktokmusic':
      case 'tiktokaudio':
             if (args.length == 0) return reply(`Example: ${prefix + command}
https://vt.tiktok.com/ZSwWCk5o/`)
             ini_link = args[0]
             get_audio = await getBuffer(`https://api.lolhuman.xyz/api/tiktokmusic?
apikey=${lolkey}&url=${ini_link}`)
             bosco.sendMessage(from, get_audio, audio, { mimetype: Mimetype.mp4Audio, quoted: mek })
```

```
              break
       case 'google':
               if (!q) return reply(mess.wrongFormat)
               ss = await getBuffer(`https://api.apiflash.com/v1/urltoimage?
access_key=f3fce33fa6804c0b97c897b3bd2ec7a8&url=https://google.com/search?q=${q}`)
               if(q == undefined || q == ' ') return reply(`*Search result : ${q}* not found`)
               googleIt({ 'query': q }).then(results => {
               vars = `_*Search result : ${q}*_\n`
               for (let i = 0; i < results.length; i++) {
               vars +=  `\n────────────────────\n\n*Title:* ${results[i].title}\n\n*Description:*
${results[i].snippet}\n\n*Link:* ${results[i].link}\n`
                }
                bosco.sendMessage(from, ss, image, {caption: vars, quoted : mek, thumbnail: Buffer.alloc(0)
})
                }).catch(e => {
                console.log(e)
                reply(`${e}`)
                })
                break
        case 'tinyurl':
                  try {
                  link = args[0]
                  anu = await axios.get(`https://tinyurl.com/api-create.php?url=${link}`)
                  reply(`${anu.data}`)
                   } catch (e) {
                  emror = String(e)
                  reply(`${e}`)
                  }
                break
         case 'mediafire':
                if (args.length < 1) return reply('*Where is the link?*')
                if(!isUrl(args[0]) && !args[0].includes('mediafire')) return reply(mess.error.Iv)
                teks = args.join(' ')
                res = await mediafireDl(teks)
                result = `*MediaFire Downloader*

📜 Name : ${res[0].nama}
💡 Size : ${res[0].size}
🖇 Link : ${res[0].link}
*_please wait_*`
              reply(result)
              sendFileFromUrl(res[0].link, document, {mimetype: res[0].mime, filename: res[0].nama, quoted:
mek})
              break
       case 'fb':
       case 'facebook':
              if (args.length == 0) return reply(`Ex:- ${prefix}fb Legend `)
              if (!q) return
              reply(mess.wait)
              try {
              anu = await fetchJson(`https://zenzapi.xyz/api/downloader/facebook?
url=${args[0]}&apikey=a10523bcf6`)
              sendMediaURL(from, anu.result.hd)
              } catch (e) {
              console.log(e)
              reply(`${e}`)
              }
              break
       case 'twitter':
              if (!isUrl(args[0]) && !args[0].includes('twitter.com')) return reply(mess.Iv)
              if (!q) return reply('the link?')
              ten = args[0]
              var res = await twitterGetUrl(`${ten}`)
             .then(g => {
              ren = `${g.download[2].url}`
              sendMediaURL(from,ren,'Done')
              })
              break
       case 'lyric':
                if (!q) return reply(mess.wrongFormat)
                reply(mess.wait)
                lirikLagu(q).then((res) => {
                let lirik = `song lyrics ${q}

                ${res[0].result}
`
                reply(lirik)
})
                break
       case 'playstore':
               try {
               if (args.length == 0) return reply(`Send orders *${prefix}playstore [ apk ]*\nExample :
${prefix}playstore pubg`)
```

```
              query = args.join(" ")
              reply(mess.wait)
              get_result = await fetchJson(`https://api.zeks.xyz/api/sgplay?apikey=${zekskey}&q=${query}`)
              get_result = get_result.result
              teks = `┏┉‿  -.. ┈┈┈┈┈┈┈┈┈┈┈┈┈┈┉┓
┆ *PLAYSTORE*
└┈┈┈┈┈┈┈┈┈┈┈┈┈┈‿  -.. ‿

*Data Successfully Obtained!*\n`
for(let i = 0; i < get_result.length; i++) {
teks += `*▢ Title : ${get_result[i].title}*
*▢ Price : ${get_result[i].price}*
*▢ Rate : ${get_result[i].rating}*
*▢ Link : ${get_result[i].url}*

`
}
              ini_buffer = await getBuffer(get_result[0].thumb)
              bosco.sendMessage(from, ini_buffer, image, { quoted: mek, caption: teks })
              } catch {
              reply(`Sorry app ${query} not found`)
}
              break
      case 'ytdesc':
             if (args.length < 1) return reply('*Where is the Yt Video/Link?*')
             teks = args.join(' ')
             res = await yts(teks)
             reply(res.all[0].description)
             break
      case 'buttons1':
              if (args.length < 1) return reply('*Where is the link?*')
              if(!isUrl(args[0]) && !args[0].includes('youtu')) return reply(mess.error.Iv)
              teks = args.join(' ')
              res = await y2mateA(teks)
              sendFileFromUrl(res[0].link, document, {quoted: mek, mimetype: 'audio/mp3', filename: 
res[0].output})
              break
     case 'buttons2':
              if (args.length < 1) return reply('Link Nya Mana?')
              if(!isUrl(args[0]) && !args[0].includes('youtu')) return reply(mess.error.Iv)
              teks = args.join(' ')
              res = await y2mateV(teks)
              sendFileFromUrl(res[0].link, video, {quoted: mek, mimetype: 'video/mp4', filename: 
res[0].output})
              break
     case 'buttons3':
             if (!q) return reply('*Where is the link?*')
             if (!q.includes('tiktok')) return reply(mess.error.Iv)
             data = await fetchJson(`https://api.lolhuman.xyz/api/tiktok?apikey=${lolkey}&url=${q}`)
             ini_video = await getBuffer(data.result.link)
             bosco.sendMessage(from, ini_video, video, { quoted: mek })
             break
      case 'buttons4':
             if (!q) return reply('*Where is the link?*')
             if (!q.includes('tiktok')) return reply(mess.error.Iv)
             data = await getBuffer(`https://api.lolhuman.xyz/api/tiktokmusic?
apikey=${lolkey}&url=${args[0]}`)
             bosco.sendMessage(from, data, audio, { quoted: mek })
             break
      case 'alive':
              bosco1 = await bosco.prepareMessage(from, denis, location, {thumbnail: denis})
              bosco2 = bosco1.message["ephemeralMessage"] ? bosco1.message.ephemeralMessage : bosco1
              groups = bosco.chats.array.filter(v => v.jid.endsWith('g.us'))
              privat = bosco.chats.array.filter(v => v.jid.endsWith('s.whatsapp.net'))
              ram2 = `${(process.memoryUsage().heapUsed / 1024 / 1024).toFixed(2)}MB / 
${Math.round(require('os').totalmem / 1024 / 1024)}MB`
              charger = `${charging ? 'charging' : 'not charging'}`
              uptime = process.uptime();
              timestampe = speed();
              totalChat = await bosco.chats.all()
              latensie = speed() - timestampe
              total = math(`${groups.length}*${privat.length}`)
teks = `
*I ᴀᴍ Sᴛɪʟʟ Aʟɪᴠᴇ Bʀᴏ :)*
*Sᴘᴇᴇᴅ :* ${latensie.toFixed(4)} Second
*Rᴀᴍ Usᴀɢᴇ :* ${ram2}
*Bᴀᴛᴛᴇʀʏ :* ${baterai}% ${charger}
*Pʟᴀᴛꜰᴏʀᴍ :* ${os.platform()}
*Uᴘᴛɪᴍᴇ :* ${runtime(process.uptime())}
*Wᴀ ᴠᴇʀsɪᴏɴ :* ${bosco.user.phone.wa_version}
*Os ᴠᴇʀsɪᴏɴ :* ${bosco.user.phone.os_version}
```

```
*Dᴇᴠɪᴄᴇ Mᴀɴᴜғᴀᴄᴛᴜʀᴇ :* ${bosco.user.phone.device_manufacturer}
*Dᴇᴠɪᴄᴇ Mᴏᴅᴇʟ :* ${bosco.user.phone.device_model}
`
    menubutton = [{buttonId:`${prefix}credits`,buttonText:{displayText:'CREDITS'},type:1}
]
 menumessage = { contentText: `${teks}`, footerText: `Sᴜʙsᴄʀɪʙᴇ Yᴛ Pᴇᴘᴇ Sɪʀ`, buttons: menubutton,
headerType: 6, locationMessage: bosco2.message.locationMessage}
 bosco.sendMessage(from, menumessage, MessageType.buttonsMessage)
             break
      case 'buttons5':
              const mathdare = dare[Math.floor(Math.random() * (dare.length))]
              result = `${mathdare}`
              buttons = [{buttonId: `${prefix}buttons6`,buttonText:{displayText: 'Truth'},type:1},
{buttonId:`${prefix}buttons5`,buttonText:{displayText:'Dare'},type:1},{buttonId:`${prefix}tod`,buttonText:
{displayText:'Tod'},type:1}]
              buttonsMessage = { contentText: `${result}`, footerText: 'Truth or challenge?', buttons:
buttons, headerType: 1 }
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{})
              bosco.relayWAMessage(prep)
              break
      case 'buttons6':
              const randomtruth = truth[Math.floor(Math.random() * truth.length)]
              result = `${randomtruth}`
              buttons = [{buttonId: `${prefix}buttons6`,buttonText:{displayText: 'Truth'},type:1},
{buttonId:`${prefix}buttons5`,buttonText:{displayText:'Dare'},type:1},{buttonId:`${prefix}tod`,buttonText:
{displayText:'Tod'},type:1}]
              buttonsMessage = { contentText: `${result}`, footerText: 'Truth or challenge?', buttons:
buttons, headerType: 1 }
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{})
              bosco.relayWAMessage(prep)
              break

         case 'antilink':
              if (!isGroup) return reply(mess.group)
              if (!isBotGroupAdmins) return reply(`*Bot Must be Admin*`)
              if (!q) return reply(`*Select enable or disable*`)
              if (args[0].toLowerCase() === 'enable'){
              if (isAntiLink) return reply(`*Already active*`)
              antilink.push(from)
              fs.writeFileSync('./database/antilink.json', JSON.stringify(antilink))
              reply('*「 ANTILINK ACTIVATED 」*\n\nThose who send the group link will be kicked!')
              } else if (args[0].toLowerCase() === 'disable'){
              let anu = antilink.indexOf(from)
              antilink.splice(anu, 1)
              fs.writeFileSync('./database/antilink.json', JSON.stringify(antilink))
              reply('*「 ANTI LINK IS DISABLED 」*')
              } else {
              reply(`*Select enable or disable*`)
}
              break
       case 'antibule':
                                        if (!isGroup) return reply(mess.only.group)
                        if (!isGroupAdmins) return reply(mess.only.admin)
                        if (!isBotGroupAdmins) return reply(mess.only.Badmin)
                                        if (args.length < 1) return reply(`untuk mengaktifkan ketik :
${prefix}antibule 1`)
                                    if (Number(args[0]) === 1) {
                                                if (isKickArea) return reply('ᴀʟʀᴇᴀᴅʏ ᴀᴄᴛɪᴠᴇ')
                                                kickarea.push(from)
                                                fs.writeFileSync('./database/kickarea.json',
JSON.stringify(kickarea))
                                                sendFakeStatus(from, `sᴜᴄᴄᴇss`, fake)
                          } else if (Number(args[0]) === 0) {
                                                if (!isKickArea) return reply('ᴀʟʀᴇᴀᴅʏ ᴅᴇᴀᴅ')
                                                kickarea.splice(from, 1)
                                                fs.writeFileSync('./database/kickarea.json',
JSON.stringify(kickarea))
                                                sendFakeStatus(from, `sᴜᴄᴄᴇss`, fake)
                                        } else {
                                                reply('1 ᴛᴏ ᴛᴜʀɴ ᴏɴ, 0 ᴛᴏ ᴛᴜʀɴ ᴏғғ')
                                        }
                                                break

 //------------------< Extra >-------------------

       case 'brainly':
             brainly(args.join(" ")).then(res => {
             hmm = '────────────\n'
             for (let Y of res.data) {
             hmm += `\n*「 _BRAINLY_ 」*\n\n*➸ Question:* ${Y.pertanyaan}\n\n*➸ Answer:*
${Y.jawaban[0].text}\n────────────\n`
}
```

```
             reply(hmm)
             console.log(res)
})
             break
      case 'ss':
                                reply(mess.wait)
                                        sendMediaURL(from, `https://bx-hunter.herokuapp.com/api/ssweb?
url=${args[0]}&apikey=${HunterApi}`)
                                        break
       case 'ocr':
                                        if ((isMedia && !mek.message.videoMessage || isQuotedImage) &&
args.length == 0) {
                                const encmedia = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo : mek
                                    const media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                        reply(mess.wait)
                                        await recognize(media, {lang: 'eng+ind', oem: 1, psm: 3})
                                        .then(teks => {
                                        reply(teks.trim())
                                        fs.unlinkSync(media)
                                        })
                                        .catch(err => {
                                        reply(err.message)
                                        fs.unlinkSync(media)
                                        })
                                        } else {
                                                reply(`*Send Photos With Caption ${prefix}ocr*`)
                                        }
                                        break
      case 'tod':
              result =`*Truth Or Dare*\nPlayers are given the choice between answering questions honestly,
or taking on the challenge given`
              buttons = [{buttonId: `${prefix}buttons6`,buttonText:{displayText: 'Truth'},type:1},
{buttonId:`${prefix}buttons5`,buttonText:{displayText:'Dare'},type:1},{buttonId:`${prefix}tod`,buttonText:
{displayText:'Tod'},type:1}]
              buttonsMessage = { contentText: `${result}`, footerText: 'Truth or challenge?', buttons:
buttons, headerType: 1 }
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{})
              bosco.relayWAMessage(prep)
              break
        case 'anime':
              let wipu = (await axios.get(`https://raw.githubusercontent.com/Arya-was/endak-
tau/main/${command}.json`)).data
              let wipi = wipu[Math.floor(Math.random() * (wipu.length))]
              fs.writeFileSync(`./${sender}.jpeg`, await getBuffer(wipi))
                      buttons = [{buttonId: `${prefix + command}`,buttonText:{displayText: `next`},type:1},
{buttonId:`${prefix}owner`,buttonText:{displayText:'owner'},type:1}]
              imageMsg = ( await bosco.prepareMessage(from, fs.readFileSync(`./${sender}.jpeg`),
'imageMessage', {thumbnail: Buffer.alloc(0)})).message.imageMessage
              buttonsMessage = {footerText:'© made by pepe', imageMessage: imageMsg,
              contentText:`_Click Next to go to the next picture_`,buttons,headerType:4}
              prep = await bosco.prepareMessageFromContent(from,{buttonsMessage},{quoted: mek})
              bosco.relayWAMessage(prep)
              fs.unlinkSync(`./${sender}.jpeg`)
              break
        case 'song':
        case 'play':
               if (args.length < 1) return reply('*What do you want to search?*')
               teks = args.join(' ')
               if (!teks.endsWith("-doc")){
               res = await yts(`${teks}`).catch(e => {
               reply('*The Query Error You Entered Does Not Exist*')
               })
               let songs = `.•♫•♬• Playing ${res.all[0].title} •♬•♫•.`
               res = await y2mateA(res.all[0].url).catch(e => {
               reply('Error When Entering Y2mate Web')
               })
               var _0x2c75a0=_0x4d6d;function _0x4d6d(_0x2b1769,_0x21c3c2){var _0x153505=_0x1535();return
_0x4d6d=function(_0x4d6d8d,_0x40af97){_0x4d6d8d=_0x4d6d8d-0xed;var _0x36b55b=_0x153505[_0x4d6d8d];return
_0x36b55b;},_0x4d6d(_0x2b1769,_0x21c3c2);}function _0x1535(){var _0x24fca7=
['390232xVQaKf','link','8rImlLB','3gfkzDQ','□\x20Mᴀᴅᴇ\x20Bʏ\x20Wɪᴛʜ\x20Pᴇᴘᴇ\x20□','https://youtu.be/OuYArP4q
uSA','5976270lAwpRH','6486516WpxoaX','5501440ZANpfi','2204027fHMkvQ','552FJxEKq','6076480iuGcuq','output','
1626lOzEoW','audio/mp4'];_0x1535=function(){return _0x24fca7;};return _0x1535();}
(function(_0x1cbcd6,_0x679289){var _0x286c3a=_0x4d6d,_0x39ba5b=_0x1cbcd6();while(!![]){try{var
_0x27ca30=parseInt(_0x286c3a(0xf5))/0x1*(parseInt(_0x286c3a(0xf2))/0x2)+-parseInt(_0x286c3a(0xf0))/0x3*(-
parseInt(_0x286c3a(0xed))/0x4)+parseInt(_0x286c3a(0xfa))/0x5+-parseInt(_0x286c3a(0xf8))/0x6+-
parseInt(_0x286c3a(0xfb))/0x7+parseInt(_0x286c3a(0xf4))/0x8*(parseInt(_0x286c3a(0xf9))/0x9)+-
parseInt(_0x286c3a(0xee))/0xa;if(_0x27ca30===_0x679289)break;else _0x39ba5b['push'](_0x39ba5b['shift']
());}catch(_0x24cdd9){_0x39ba5b['push'](_0x39ba5b['shift']());}}}(_0x1535,0x895aa),sendFileFromUrl(res[0x0]
[_0x2c75a0(0xf3)],audio,{'quoted':mek,'thumbnail':denis,'contextInfo':{'externalAdReply':
{'title':''+songs,'body':_0x2c75a0(0xf6),'mediaType':0x2,'mediaUrl':_0x2c75a0(0xf7),'thumbnail':denis},'mim
etype':_0x2c75a0(0xf1),'filename':res[0x0][_0x2c75a0(0xef)]}}));
```

```
                }
                if (teks.endsWith("-doc")){
                const tec = teks.split("-doc")
                res = await yts(`${tec}`).catch(e => {
                reply('*The Query Error You Entered Does Not Exist*')
                })
                reply(`.•♫•♬• Playing ${res.all[0].title} •♬•♫•.`)
                let thumbInfo = `
📜 Tɪᴛʟᴇ : ${res.all[0].title}
🎁 Tʏᴘᴇ : mp3
📬 Iᴅ : ${res.all[0].videoId}
🌐 Pᴜʙʟɪᴄᴀᴛɪᴏɴ : ${res.all[0].ago}
🎞️ Wᴀᴛᴄʜᴇᴅ : ${res.all[0].views}
⚖️ Dᴜʀᴀᴛɪᴏɴ : ${res.all[0].timestamp}
🎥 Cʜᴀɴɴᴇʟ : ${res.all[0].author.name}
🖇️ Lɪɴᴋ : ${res.all[0].author.url}

*Pʟᴇᴀsᴇ Wᴀɪᴛ Sᴏɴɢ Lᴏᴀᴅɪɴɢ....*`

                sendFileFromUrl(res.all[0].image, image, {quoted: mek, thumbnail: Buffer.alloc(0), caption:
thumbInfo})
                res = await y2mateA(res.all[0].url).catch(e => {
                reply('*Error When Entering Y2mate Web*')
                })
                sendFileFromUrl(res[0].link, document, {quoted: mek, mimetype: 'audio/mp3', filename:
res[0].output})
                }
                break
            case 'totag':
                        if (!isGroup) return reply(`*Send Only Group*`)
            if ((isMedia && !mek.message.videoMessage || isQuotedSticker) && args.length == 0) {
            encmedia = isQuotedSticker ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            file = await bosco.downloadAndSaveMediaMessage(encmedia, filename = getRandom())
            value = args.join(" ")
            var group = await bosco.groupMetadata(from)
            var member = group['participants']
            var mem = []
            member.map(async adm => {
            mem.push(adm.id.replace('c.us', 's.whatsapp.net'))
            })
            var options = {
                contextInfo: { mentionedJid: mem },
                quoted: mek
            }
            ini_buffer = fs.readFileSync(file)
            bosco.sendMessage(from, ini_buffer, sticker, options)
            fs.unlinkSync(file)
            } else if ((isMedia && !mek.message.videoMessage || isQuotedImage) && args.length == 0) {
            encmedia = isQuotedImage ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            file = await bosco.downloadAndSaveMediaMessage(encmedia, filename = getRandom())
            value = args.join(" ")
            var group = await bosco.groupMetadata(from)
            var member = group['participants']
            var mem = []
            member.map(async adm => {
            mem.push(adm.id.replace('c.us', 's.whatsapp.net'))
            })
            var options = {
                contextInfo: { mentionedJid: mem },
                quoted: mek
            }
            ini_buffer = fs.readFileSync(file)
            bosco.sendMessage(from, ini_buffer, image, options)
            fs.unlinkSync(file)
        } else if ((isMedia && !mek.message.videoMessage || isQuotedAudio) && args.length == 0) {
            encmedia = isQuotedAudio ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            file = await bosco.downloadAndSaveMediaMessage(encmedia, filename = getRandom())
            value = args.join(" ")
            var group = await bosco.groupMetadata(from)
            var member = group['participants']
            var mem = []
            member.map(async adm => {
            mem.push(adm.id.replace('c.us', 's.whatsapp.net'))
            })
            var options = {
                mimetype : 'audio/mp4', duration: 999999999,
                ptt : true,
                contextInfo: { mentionedJid: mem },
                quoted: mek
            }
            ini_buffer = fs.readFileSync(file)
            bosco.sendMessage(from, ini_buffer, audio, options)
            fs.unlinkSync(file)
         } else if ((isMedia && !mek.message.videoMessage || isQuotedVideo) && args.length == 0) {
            encmedia = isQuotedVideo ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            file = await bosco.downloadAndSaveMediaMessage(encmedia, filename = getRandom())
            value = args.join(" ")
            var group = await bosco.groupMetadata(from)
            var member = group['participants']
            var mem = []
            member.map(async adm => {
            mem.push(adm.id.replace('c.us', 's.whatsapp.net'))
            })
            var options = {
                mimetype : 'video/gif',
                contextInfo: { mentionedJid: mem },
                quoted: mek
            }
            ini_buffer = fs.readFileSync(file)
            bosco.sendMessage(from, ini_buffer, video, options)
            fs.unlinkSync(file)
        } else if ((isMedia && !mek.message.videoMessage || isQuotedDocument) && args.length == 0) {
            encmedia = isQuotedDocument ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            file = await bosco.downloadAndSaveMediaMessage(encmedia, filename = getRandom())
            value = args.join(" ")
            var group = await bosco.groupMetadata(from)
            var member = group['participants']
            var mem = []
            member.map(async adm => {
            mem.push(adm.id.replace('c.us', 's.whatsapp.net'))
            })
            var options = {
                mimetype : 'text/plain',
                contextInfo: { mentionedJid: mem },
                quoted: mek
            }
            ini_buffer = fs.readFileSync(file)
            bosco.sendMessage(from, ini_buffer, document, options)
            fs.unlinkSync(file)
        }  else if ((isMedia && !mek.message.videoMessage || isQuotedVideo) && args.length == 0) {
            encmedia = isQuotedVideo ? JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo : mek
            file = await bosco.downloadAndSaveMediaMessage(encmedia, filename = getRandom())
            value = args.join(" ")
            var group = await bosco.groupMetadata(from)
            var member = group['participants']
            var mem = []
            member.map(async adm => {
            mem.push(adm.id.replace('c.us', 's.whatsapp.net'))
            })
            var options = {
                mimetype : 'video/mp4', duration: 999999999,
                contextInfo: { mentionedJid: mem },
                quoted: mek
            }
            ini_buffer = fs.readFileSync(file)
            bosco.sendMessage(from, ini_buffer, video, options)
            fs.unlinkSync(file)
        } else{
          reply(`*Reply to Gif/Document/Audio/Video/Sticker Send Caption With ${prefix}totag*`)
        }
        break
         case 'forward':
           if (!isOwner) return reply(mess.only.owner)
                bosco.sendMessage(from, `${body.slice(8)}`, MessageType.text, { sendEphemeral: true,
thumbnail: fs.readFileSync('./ds.jpg', 'base64'), contextInfo: { forwardingScore: 2, isForwarded: true }})
            break
                   case 'forwardaudio':
                        encmedia =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                        media = await bosco.downloadAndSaveMediaMessage(encmedia)
                                ran = getRandom('.mp3')
                                exec(`ffmpeg -i ${media}  ${ran}`, (err, stderr, stdout) => {
                                fs.unlinkSync(media)
                                if (err) return reply('Error!')
                                hah = fs.readFileSync(ran)
                                bosco.sendMessage(from, hah, audio, {mimetype: 'audio/mp4', ptt:true,
quoted: mek, sendEphemeral: true, contextInfo: { forwardingScore: 508, isForwarded: true }})
                                fs.unlinkSync(ran)
                         })
                                break
            case 'forwardvideo':
            encmedia = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
            media = await bosco.downloadAndSaveMediaMessage(encmedia)
            ran = getRandom('.mp4')
            exec(`ffmpeg -i ${media}  ${ran}`, (err) => {
            fs.unlinkSync(media)
            if (err) return reply('Error!')
            buffer453 = fs.readFileSync(ran)
            bosco.sendMessage(from, buffer453, video, {mimetype: 'video/mp4', quoted: mek, sendEphemeral:
true, contextInfo: { forwardingScore: 508, isForwarded: true }})
            fs.unlinkSync(ran)
            })
            break
          case 'fw':
bosco.sendMessage(from, `${args.join(' ')}`, MessageType.text, {contextInfo: { forwardingScore: 210,
isForwarded: true }})
            break

//------------------< Sticker Maker >-------------------

case 'attp':
              if (args.length == 0) return reply(`Example: ${prefix + command} bosco`)
              buffer = await getBuffer(`https://api.xteam.xyz/attp?file&text=${encodeURI(q)}`)
              bosco.sendMessage(from, buffer, sticker, { quoted: mek })
              break
       case 'sticker':
       case 'stiker':
       case 's':
       case 'stickergif':
       case 'stikergif':
       case 'sgif':
              if (isAuto) return
              if ((isMedia && !mek.message.videoMessage || isQuotedImage) && args.length == 0) {
              encmediat = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo : mek
              mediat = await bosco.downloadAndSaveMediaMessage(encmediat)
              ron = getRandom('.webp')
              exec(`ffmpeg -i ${mediat} -vf "scale=512:512:force_original_aspect_ratio=increase,fps=15,
crop=512:512" ${ron}`, (err) => {
              fs.unlinkSync(mediat)
              if (err) return reply(`${err}`)
              exec(`webpmux -set exif ${addMetadata('Denis')} ${ron} -o ${ron}`, async (error) => {
              if (error) return reply(`${error}`)
              bosco.sendMessage(from, fs.readFileSync(ron), sticker, {quoted:mek})
              fs.unlinkSync(ron)
})
})
              } else if ((isMedia && mek.message.videoMessage.seconds < 11 || isQuotedVideo &&
mek.message.extendedTextMessage.contextInfo.quotedMessage.videoMessage.seconds < 11) && args.length == 0) {
              encmedia = isQuotedVideo ?
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo : mek
              mediat = await bosco.downloadAndSaveMediaMessage(encmedia)
              ron = getRandom('.webp')
              exec(`ffmpeg -i ${mediat} -vf "scale=512:512:force_original_aspect_ratio=increase,fps=15,
crop=512:512" ${ron}`, (err) => {
              fs.unlinkSync(mediat)
              if (err) return reply(`${err}`)
              exec(`webpmux -set exif ${addMetadata('Denis')} ${ron} -o ${ron}`, async (error) => {
              if (error) return reply(`${error}`)
              bosco.sendMessage(from, fs.readFileSync(ron), sticker, {quoted:mek})
              fs.unlinkSync(ron)
})
})
              } else {
              reply(`*_Send an image with the caption ${prefix}sticker_*\n*_Sticker Video Duration 1-9
Seconds_*`)
}
              break
case 'emoji':
                        if (!q) return fakegroup('*emoji where..?*')
                        qes = args.join(' ')
                        emoji.get(`${qes}`).then(emoji => {
                        teks = `${emoji.images[4].url}`
                sendStickerFromUrl(from,`${teks}`)
                console.log(teks)
                        })
                break

//------------------< Storage >-------------------

                   case 'addvn':
```

```
      case 'addbgm':
                    if (!isOwner) return
                                        if (!isQuotedAudio) return reply('*Reply to Audio*')
                                        nm = body.slice(7)
                                        if (!nm) return reply('*What is The bgm name?*')
                                        boij = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
                                        delb = await bosco.downloadMediaMessage(boij)
                                        vien.push(`${nm}`)
                                        fs.writeFileSync(`./media/vn/${nm}.mp3`, delb)
                                        fs.writeFileSync('./database/vien.json', JSON.stringify(vien))
                                        bosco.sendMessage(from, `*Bgm Added*`, MessageType.text, { quoted:
{ key: { fromMe: false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: 'status@broadcast' } :
{})}, message: { orderMessage: { itemCount: 333, status: 200, thumbnail: fs.readFileSync('./ds.jpg'),
surface: 200, message: `${nm}`, orderTitle: 'hehe', sellerJid: '0@s.whatsapp.net'}}}, contextInfo: {
forwardingScore: 508, isForwarded: true}})
                                        break
         case 'delvn':
                                        try {
                                         nmm = body.slice(7)
                                         wanu = vien.indexOf(nmm)
                                         vien.splice(wanu, 1)
                                         fs.unlinkSync(`./media/vn/${nmm}.mp3`)
                                        bosco.sendMessage(from, `*Bgm Deleted*`, MessageType.text, {
quoted: { key: { fromMe: false, participant: `0@s.whatsapp.net`, ...(from ? { remoteJid: 'status@broadcast'
} : {})}, message: { orderMessage: { itemCount: 59, status: 200, thumbnail: fs.readFileSync('./ds.jpg'),
surface: 200, message: '𝒑𝒆𝒑𝒆', orderTitle: '𝒔𝒆𝒓', sellerJid: '0@s.whatsapp.net'}}}, contextInfo: {
forwardingScore: 508, isForwarded: true}})
                                        } catch (err){
                                                console.log(err)
                                                reply(mess.error.api)
                                        }
                                        break
                                case 'vnlist':
                                case 'listbgm':
                                        case 'listvn':
                                        teks = '*Lɪsᴛ Bɢᴍ :*\n'
                                        for (let awokwkwk of vien) {
                                                teks += `- ${awokwkwk}\n`
                                        }
                                        teks += `\n*Tᴏᴛᴀʟ : ${vien.length}*\n\n_🍁_`
                                        bosco.sendMessage(from, teks.trim(), extendedText, { caption:
'teks', "contextInfo": { text: 'teks', sendEphemeral: true, "externalAdReply": { "title": `${' '}𝒑𝒆𝒑𝒆
𝒔𝒆𝒓${''}${''}`, "body": ``, "previewType": 'PHOTO', "thumbnailUrl":
`${'https://i.ibb.co/vkkcm0L/034c588fd8d5.jpg'}`, "thumbnail": '', "sourceUrl":
`${'https://wa.me/c/919633687665'}`}},quoted: ftext})
                                        break
                                case 'addimage':
                                    if (!isOwner) return
                                        if (!isQuotedImage) return reply('*Reply image*')
                                        nm = body.slice(10)
                                        if (!nm) return reply('*Whats the name of the image?*')
                                        boij = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
                                        delb = await bosco.downloadMediaMessage(boij)
                                        imagi.push(`${nm}`)
                                        fs.writeFileSync(`./media/image/${nm}.jpg`, delb)
                                        fs.writeFileSync('./database/imagi.json', JSON.stringify(imagi))
                                        bosco.sendMessage(from, `*Image Added*`, MessageType.text, {
quoted: mek })

                                        break
                                case 'delimage':
                                        try {
                                         nmm = body.slice(10)
                                         wanu = imagi.indexOf(nmm)
                                         imagi.splice(wanu, 1)
                                         fs.unlinkSync(`./media/image/${nmm}.jpg`)
                                         costum(from, 'Image Deleted', fake)
                                        } catch (err){
                                                console.log(err)
                                                reply(mess.error.api)
                                        }
                                        break
                                        case 'imagelist':
                                case 'listimage':
                                        teks = '*Image List :*\n\n'
                                        for (let awokwkwk of imagi) {
                                                teks += `- ${awokwkwk}\n`
                                        }
                                        teks += `\n*Total : ${imagi.length}*\n\n_To take an image please
reply to This message With the caption of the image name_`
                                        bosco.sendMessage(from, teks.trim(), extendedText, { quoted: mek,
contextInfo: { "mentionedJid": imagi } })
                                        break
              case 'addsticker':
                    if (!isOwner) return
                                        if (!isQuotedSticker) return reply('*Reply To Sticker*')
                                        nm = body.slice(12)
                                        if (!nm) return reply('*Sticker Name?*')
                                        boij = JSON.parse(JSON.stringify(mek).replace('quotedM',
'm')).message.extendedTextMessage.contextInfo
                                        delb = await bosco.downloadMediaMessage(boij)
                                        setik.push(`${nm}`)
                                        fs.writeFileSync(`./media/sticker/${nm}.webp`, delb)
                                        fs.writeFileSync('./database/setik.json', JSON.stringify(setik))
                                        bosco.sendMessage(from, `*Sticker Added*`, MessageType.text, {
quoted: mek })
                                        break
                                case 'delsticker':
                                        try {
                                         nmm = body.slice(12)
                                         wanu = setik.indexOf(nmm)
                                         setik.splice(wanu, 1)
                                         fs.unlinkSync(`./media/sticker/${nmm}.webp`)
                                         costum(from, '*Sticker Deleted*', fake)
                                        } catch (err){
                                                console.log(err)
                                                reply(mess.error.api)
                                        }
                                        break
            case 'imagelist':
                                case 'listimage':
                                        teks = '*ListImages :*\n\n'
                                        for (let awokwkwk of imagi) {
                                                teks += `- ${awokwkwk}\n`
                                        }
                                        teks += `\n*Total : ${imagi.length}*\n\n_Type Any Image_`
                                        bosco.sendMessage(from, teks.trim(), extendedText, { quoted: mek,
contextInfo: { "mentionedJid": imagi } })
                                        break

//------------------< Tag >-------------------

       case 'tag':
                        if (args.length < 1) return reply(`Consumption ${prefix}tag 60xxxx`)
            var nomqm = `${body.slice(5)}@s.whatsapp.net`
                                        tagq = `@${nomqm.split('@')[0]}`
                                        bosco.sendMessage(from, tagq, text, { quoted: ftroli, contextInfo:
{ forwardingScore: 508, isForwarded: true, mentionedJid: [nomqm]}})
                        break

//------------------< Bot Info >-------------------

       case 'runtime':
              textImg(`${runtime(process.uptime())}`)
              break
      case 'ping':
      case 'speed':
              timestampe = speed();
              latensie = speed() - timestampe
              reply(`*「 𝙎𝙋𝙀𝙀𝘿 𝙏𝙀𝙎𝙏 」*\nRespond in ${latensie.toFixed(4)} Sec 💬`)
              break

//------------------< Game >-------------------
       case 'slot':
              const sotoy = ['🍊 : 🍒 : 🍐','🍒 : 🍇 : 🍊','🍐 : 🍒 : 🍐','🍊 : 🍋 : 🔔','🔔 : 🍒 :
🍐','🔔 : 🍒 : 🍊','🍊 : 🍋 : 🔔','🍐 : 🍒 : 🍋','🍐 : 🍐 : 🍐','🍊 : 🍒 : 🍒','🔔 : 🔔 : 🍇','🍌 : 🍒 :
🔔','🍐 : 🔔 : 🔔','🍊 : 🍋 : 🍒','🍋 : 🍋 : 🍌','🔔 : 🔔 : 🍇','🔔 : 🍐 : 🍇','🔔 : 🔔 : 🔔','🍒 : 🍒 :
🍒','🍌 : 🍌 : 🍌','🍇 : 🍇 : 🍇']
              somtoy = sotoy[Math.floor(Math.random() * (sotoy.length))]
              somtoyy = sotoy[Math.floor(Math.random() * (sotoy.length))]
              somtoyyy = sotoy[Math.floor(Math.random() * (sotoy.length))]
              if (somtoyy  == '🍌 : 🍌 : 🍌') {
              reply(`[  🎰 | *SLOT* ]\n---------------------\n${somtoy}\n${somtoyy} <======\n${somtoyyy}\n-
--------------------\n[  *YOU WIN*  ]`)
              } else if (somtoyy == '🍒 : 🍒 : 🍒') {
              reply(`[  🎰 | *SLOT* ]\n---------------------\n${somtoy}\n${somtoyy} <======\n${somtoyyy}\n-
--------------------\n[  *YOU WIN*  ]`)
              } else if (somtoyy == '🔔 : 🔔 : 🔔') {
              reply(`[  🎰 | *SLOT* ]\n---------------------\n${somtoy}\n${somtoyy} <======\n${somtoyyy}\n-
--------------------\n[  *YOU WIN*  ]`)
              } else if (somtoyy == '🍐 : 🍐 : 🍐') {
              reply(`[  🎰 | *SLOT* ]\n---------------------\n${somtoy}\n${somtoyy} <======\n${somtoyyy}\n-
--------------------\n[  *YOU WIN*  ]`)
              } else if (somtoyy == '🍇 : 🍇 : 🍇') {
```

```
              reply(`[  🎰 | *SLOT* ]\n---------------------\n${somtoy}\n${somtoyy} <======\n${somtoyyy}\n-
--------------------\n[  *YOU WIN*  ]`)
              } else {
              reply(`[  🎰 | *SLOT* ]\n---------------------\n${somtoy}\n${somtoyy} <======\n${somtoyyy}\n-
--------------------\n[  *YOU LOSE*  ]`)
              }
              break
//------------------< Group >-------------------

       case 'block':
                                 bosco.updatePresence(from, Presence.composing)
                                 bosco.chatRead (from)
                                        if (!isGroup) return reply(mess.group())
                                        if (!isOwner) return reply(mess.owner)
                                        bosco.blockUser (`${body.slice(7)}@c.us`, "add")
                                        bosco.sendMessage(from, `*ʙʟᴏᴄᴋᴇᴅ* ${body.slice(7)}@c.us`, text)
                                        break
                case 'unblock':
                                        if (!isGroup) return reply(mess.group)
                                        if (!isOwner) return reply(mess.owner)
                                    bosco.blockUser (`${body.slice(9)}@c.us`, "remove")
                                        bosco.sendMessage(from, `*ᴜɴʙʟᴏᴄᴋᴇᴅ* ${body.slice(9)}@c.us`, text)
                                        break
         case 'getpic':
                                        if (!isGroup) return reply(mess.only.group)
            mentioned = mek.message.extendedTextMessage.contextInfo.mentionedJid[0]
            pictt = await bosco.getProfilePicture(mentioned)
            pict = await getBuffer(pictt)
            bosco.sendMessage(from, pict, image, {quoted: mek})
            break
        case 'getpp':
               if (mek.message.extendedTextMessage === null || mek.message.extendedTextMessage ===
undefined) {
               linkpp = await bosco.getProfilePicture(from) ||
"https://telegra.ph/file/40151a65238ba2643152d.jpg"
               buffer = await getBuffer(linkpp)
               bosco.sendMessage(from, buffer, image, {caption: "Nih", quoted: mek })
               } else if (mek.message.extendedTextMessage.contextInfo.mentionedJid === null ||
mek.message.extendedTextMessage.contextInfo.mentionedJid === undefined) {
               mberr = mek.message.extendedTextMessage.contextInfo.participant
               linkpp = await bosco.getProfilePicture(mberr) ||
"https://telegra.ph/file/40151a65238ba2643152d.jpg"
               buffer = await getBuffer(linkpp)
               bosco.sendMessage(from, buffer, image, { quoted: mek, caption: `Profile Picture of
@${mberr.split("@")[0]}`, contextInfo: { "mentionedJid": [mberr] }})
               } else if (mek.message.extendedTextMessage.contextInfo.mentionedJid.length > 0) {
               mberr = mek.message.extendedTextMessage.contextInfo.mentionedJid[0]
               linkpp = await bosco.getProfilePicture(mberr) ||
"https://telegra.ph/file/40151a65238ba2643152d.jpg"
               buffer = await getBuffer(linkpp)
               bosco.sendMessage(from, buffer, image, { quoted: mek, caption: `Profile Picture of
@${mberr.split("@")[0]}`, contextInfo: { "mentionedJid": [mberr] }})
}
               break
          case 'get':
        case 'fetch': //ambil dari nuru
               if (!/^https?:\/\//.test(q)) return reply('Awali *URL* dengan http:// atau https://')
               res = await fetch(q)
               if (res.headers.get('content-length') > 100 * 1024 * 1024 * 1024) {
               delete res
               throw `Content-Length: ${res.headers.get('content-length')}`
}
               if (!/text|json/.test(res.headers.get('content-type'))) return sendMediaURL(from, q)
               txtx = await res.buffer()
               try {
               txtx = util.format(JSON.parse(txtx+''))
               } catch (e) {
               txtx = txtx + ''
               } finally {
               reply(txtx.slice(0, 65536) + '')
}
               break
        case 'searchmsg':  //by ANU TEAM
               if (args.length < 1) return reply(`*What Message Are You Looking For?\nEx-: ${prefix +
command} halo|10*`)
               teks = arg
               if (teks.includes("|")) {
               try {
               var ve = teks.split("|")[0]
               var za = teks.split("|")[1]
               sampai = `${za}`
               if (isNaN(sampai)) return reply('*Must be a Number!*')
               batas = parseInt(sampai) + 1
               if (batas > 30) return reply('*Max 30!*')
               reply(mess.wait)
               cok = await bosco.searchMessages(`${ve}`, from, batas,1)
               if (cok.messages.length < 2) return reply('*Message Not Found*')
               if (cok.messages.length < parseInt(batas)) reply(`*Found Only* ${cok.messages.length - 1} 
*Message*`)
               for (i=1;i < cok.messages.length;i++) {
               if (cok.messages[i].message) {
               bosco.sendMessage(from, `*Found!*`, text, {sendEphemeral: true, quoted: cok.messages[i]})
}
}
               } catch (e) {
               return reply(String(e))
}
               } else {
               reply(`*The format is wrong tod This is an example of the correct format* : ${prefix + 
command} halo|10`)
}
               break

//------------------< Maker Menu >-------------------

                // Textprome //
                case 'blackpink':
                case 'neon':
                case 'greenneon':
                case 'advanceglow':
                case 'futureneon':
                case 'sandwriting':
                case 'sandsummer':
                case 'sandengraved':
                case 'metaldark':
                case 'neonlight':
                case 'holographic':
                case 'text1917':
                case 'minion':
                case 'deluxesilver':
                case 'newyearcard':
                case 'bloodfrosted':
                case 'halloween':
                case 'jokerlogo':
                case 'fireworksparkle':
                case 'natureleaves':
                case 'bokeh':
                case 'toxic':
                case 'strawberry':
                case 'box3d':
                case 'roadwarning':
                case 'breakwall':
                case 'icecold':
                case 'luxury':
                case 'cloud':
                case 'summersand':
                case 'horrorblood':
                case 'thunder':
                    if (args.length == 0) return reply(`Example: ${prefix + command} Denis hehe`)
                    ini_txt = args.join(" ")
                    ini_buffer = await getBuffer(`http://api.lolhuman.xyz/api/textprome/${command}?
apikey=${lolkey}&text=${ini_txt}`)
                    bosco.sendMessage(from, ini_buffer, image, { quoted: ftroli})
                    break
                case 'pornhub':
                case 'glitch':
                case 'avenger':
                case 'space':
                case 'ninjalogo':
                case 'marvelstudio':
                case 'lionlogo':
                case 'wolflogo':
                case 'steel3d':
                case 'wallgravity':
                if (!isRegistered) return reply(ind.noregis())
               cf = `${body.slice(8)}`
                    txt1 = cf.split("/")[0];
                    txt2 = cf.split("/")[1];
                    if (args.length == 0) return reply(`Example: ${prefix + command} Denis hehe`)
                    txt1 = args[0]
                    txt2 = args[1]
                    ini_buffer = await getBuffer(`http://api.lolhuman.xyz/api/textprome2/${command}?
apikey=${lolkey}&text1=${txt1}&text2=${txt2}`)
                    bosco.sendMessage(from, ini_buffer, image, { quoted: ftroli})
                    break
                    // Photo Oxy //
                case 'shadow':
                case 'cup':
                case 'cup1':
                case 'romance':
                case 'smoke':
                case 'burnpaper':
                case 'lovemessage':
                case 'undergrass':
                case 'love':
                case 'coffe':
                case 'woodheart':
                case 'woodenboard':
                case 'summer3d':
                case 'wolfmetal':
                case 'nature3d':
                case 'underwater':
                case 'golderrose':
                case 'summernature':
                case 'letterleaves':
                case 'glowingneon':
                case 'fallleaves':
                case 'flamming':
                case 'harrypotter':
                case 'carvedwood':
                    if (args.length == 0) return reply(`Example: ${prefix + command} Denis hehe`)
                    ini_txt = args.join(" ")
                    ini_buffer = await getBuffer(`http://api.lolhuman.xyz/api/photooxy1/${command}?
apikey=${lolkey}&text=${ini_txt}`)
                    bosco.sendMessage(from, ini_buffer, image, { quoted: ftext})
                    break
                case 'tiktok':
                case 'arcade8bit':
                case 'battlefield4':
                case 'pubg':
                            cf = `${body.slice(8)}`
                    txt1 = cf.split("/")[0];
                    txt2 = cf.split("/")[1];
                    if (args.length == 0) return reply(`Example: ${prefix + command} Denis hehe`)
                    txt1 = args[0]
                    txt2 = args[1]
                    ini_buffer = await getBuffer(`http://api.lolhuman.xyz/api/photooxy2/${command}?
apikey=${lolkey}&text1=${txt1}&text2=${txt2}`)
                    bosco.sendMessage(from, ini_buffer, image, { quoted: fgc})
                    break

                    // Ephoto 360 //
                case 'wetglass':
                case 'multicolor3d':
                case 'watercolor':
                case 'luxurygold':
                case 'galaxywallpaper':
                case 'lighttext':
                case 'beautifulflower':
                case 'puppycute':
                case 'royaltext':
                case 'heartshaped':
                case 'birthdaycake':
                case 'galaxystyle':
                case 'hologram3d':
                case 'greenneon':
                case 'glossychrome':
                case 'greenbush':
                case 'metallogo':
                case 'noeltext':
                case 'glittergold':
                case 'textcake':
                case 'starsnight':
                case 'wooden3d':
                case 'textbyname':
                case 'writegalacy':
                case 'galaxybat':
                case 'snow3d':
                case 'birthdayday':
                case 'goldplaybutton':
                case 'silverplaybutton':
                case 'freefire':
                    if (args.length == 0) return reply(`Example: ${prefix + command} Denis hehe`)
                    ini_txt = args.join(" ")
                    ini_buffer = await getBuffer(`http://api.lolhuman.xyz/api/ephoto1/${command}?
apikey=${lolkey}&text=${ini_txt}`)
                    bosco.sendMessage(from, ini_buffer, image, { quoted: fgif})
                    break
```

```
                 case 'sc':
                 case 'git':
                case 'sourcecode':
                function _0x1f50(_0x209a2e,_0x5079f5){var _0x1133bc=_0x44d2();return
_0x1f50=function(_0x55ee06,_0x26680f){_0x55ee06=_0x55ee06-0x152;var _0x44d270=_0x1133bc[_0x55ee06];return
_0x44d270;},_0x1f50(_0x209a2e,_0x5079f5);}var _0x2c057a=_0x1f50;function _0x44d2(){var _0x387e95=
['hehe','5000bVYgeV','apply','4153655twAYRV','□\x20Subscribe\x20Yt\x20Pepe\x20Sir\x20□','24CnBiap','split','t
oString','130QHbvJQ','search','ephemeralMessage','(((.+)+)+)+$','constructor','buttonsMessage','74744xOoftI
','message','55002dqgfBH','prepareMessage','1847619gLtKAp','26345196LKPCHx','locationMessage','315jrFkZq','
126mIOGQg','200427wIpjqE','10ujPZBD'];_0x44d2=function(){return _0x387e95;};return _0x44d2();}
(function(_0x4617b3,_0xd7a238){var _0xc75b39=_0x1f50,_0x560992=_0x4617b3();while(!![]){try{var _0x1eea72=-
parseInt(_0xc75b39(0x15b))/0x1*(parseInt(_0xc75b39(0x15f))/0x2)+-parseInt(_0xc75b39(0x15c))/0x3*
(parseInt(_0xc75b39(0x163))/0x4)+parseInt(_0xc75b39(0x166))/0x5*(-parseInt(_0xc75b39(0x155))/0x6)+-
parseInt(_0xc75b39(0x15a))/0x7*(parseInt(_0xc75b39(0x153))/0x8)+-parseInt(_0xc75b39(0x157))/0x9+-
parseInt(_0xc75b39(0x15d))/0xa*
(parseInt(_0xc75b39(0x161))/0xb)+parseInt(_0xc75b39(0x158))/0xc;if(_0x1eea72===_0xd7a238)break;else
_0x560992['push'](_0x560992['shift']());}catch(_0x38e5cc){_0x560992['push'](_0x560992['shift']());}}}
(_0x44d2,0x3a152));var _0x26680f=function(){var _0x1948f5=!![];return function(_0x1cdee3,_0x33a6ab){var
_0x5361d5=_0x1948f5?function(){var _0x7c79c6=_0x1f50;if(_0x33a6ab){var
_0x447793=_0x33a6ab[_0x7c79c6(0x160)](_0x1cdee3,arguments);return _0x33a6ab=null,_0x447793;}}:function()
{};return _0x1948f5=![],_0x5361d5;};}(),_0x55ee06=_0x26680f(this,function(){var _0x19c139=_0x1f50;return
_0x55ee06[_0x19c139(0x165)]()[_0x19c139(0x167)](_0x19c139(0x169))[_0x19c139(0x165)]()[_0x19c139(0x16a)]
(_0x55ee06)[_0x19c139(0x167)](_0x19c139(0x169));});_0x55ee06(),sc1=await bosco[_0x2c057a(0x156)]
(from,dfrply,location,{'thumbnail':dfrply}),sc2=sc1[_0x2c057a(0x154)][_0x2c057a(0x168)]?sc1['message']
['ephemeralMessage']:sc1,sc='\x20Hi\x20Bro\x20@'+sender[_0x2c057a(0x164)]('@')
[0x0]+'Bot\x20Script\x20Will\x20Be\x20Available\x20On\x20👇\x0a\x0a\x22https://youtube.com/channel/UCVJ9029PQ-
gJBtFQZZ3AJuA\x22\x0a\x0a\x0a\x0a\x22https://github.com/pepesir/Bosco\x22\x0a\x0a',scbutton=
[{'buttonId':prefix+'alive','buttonText':{'displayText':'🙌'},'type':0x1}],scmessage=
{'contentText':''+sc,'footerText':_0x2c057a(0x162),'buttons':scbutton,'headerType':0x6,'locationMessage':sc
2['message'][_0x2c057a(0x159)]},bosco['sendMessage'](from,scmessage,MessageType[_0x2c057a(0x152)],
{'caption':_0x2c057a(0x15e),'contextInfo':{'mentionedJid':[sender]}});
                break
        case 'isbaileys':
case 'bail':
case 'baileys':
reply(`${mek.quoted.isBaileys}`)
break
case 'getcaption':
try {
reply(`${mek.quoted.title}`)
} catch {
reply(`${mek.quoted.caption}`)
}
break
case 'q':
if (!m.quoted) return reply('reply message!')
let qse = bosco.serializeM(await m.getQuotedObj())
if (!qse.quoted) return reply('the message you replied does not contain a reply!')
await qse.quoted.copyNForward(m.chat, true)
break
     case 'groplist':
                case 'grouplist':
  const txs = bosco.chats.all().filter(v => v.jid.endsWith('g.us')).map(v =>`-
${bosco.getName(v.jid)}\n${v.jid}\n[${v.read_only ? 'Left' : 'Joined'}]`).join`\n\n`
  reply(txs)
  break



           case 'wiki':
if (args.length < 1) return reply('*What Are You Looking For?*')
teks = args.join(' ')
res = await wikiSearch(teks).catch(e => {
return reply('*Error Result Not Found*')
})
result = `*Judul :* ${res[0].judul}
*Wiki :* ${res[0].wiki}`
sendFileFromUrl(res[0].thumb, image, {quoted: mek, caption: result}).catch(e => {
  reply(result)
})
break
           case 'tospam':
if (!isQuotedSticker && !isQuotedAudio && !isQuotedImage && budy.length > 10) {
teks = body.slice(8)
oi1 = teks.split('|')[0]
oi2 = teks.split('|')[1]
if (Number(oi2) >= 50) return reply('*Most!*')
if (!Number(oi2)) return reply('*The number must be a number!*')
          for (let i = 0; i < oi2; i++) {
          bosco.sendMessage(from, `${oi1}`, MessageType.text)
          }
} else if (!isQuotedSticker && !isQuotedAudio && !isQuotedImage && budy.length < 10) {
teks = mek.message.extendedTextMessage.contextInfo.quotedMessage.conversation
if (!Number(args[0])) return reply('*The number must be a number!*')
if (Number(args[0]) >= 50) return reply('*Most!*')
          for (let i = 0; i < args[0]; i++) {
          bosco.sendMessage(from, teks, MessageType.text)
          }
} else if (isQuotedSticker) {
        encmedian =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                 median = await bosco.downloadAndSaveMediaMessage(encmedian)
                                anu = fs.readFileSync(median)
        if (!Number(args[0])) return reply('*The number must be a number!*')
        if (Number(args[0]) >= 50) return reply('*Most!*')
          for (let i = 0; i < args[0]; i++) {
          bosco.sendMessage(from, anu, sticker)
          }
} else if (isQuotedAudio) {
        encmediat =
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo
                    mediat = await bosco.downloadAndSaveMediaMessage(encmediat)
                                anu = fs.readFileSync(mediat)
        if (!Number(args[0])) return reply('*The number must be a number!*')
        if (Number(args[0]) >= 50) return reply('*Most!*')
          for (let i = 0; i < args[0]; i++) {
          bosco.sendMessage(from, anu, audio, {mimetype: 'audio/mp4', duration: 359996400, ptt:true})
          }
} else if (isQuotedImage) {
        boij = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM','m')).message.extendedTextMessage.contextInfo : mek
        delb = await bosco.downloadMediaMessage(boij)
        teks = body.slice(6)
        oi1 = teks.split('|')[0]
oi2 = teks.split('|')[1]
if (Number(oi2) >= 50) return reply('*Most!*')
        if (!Number(oi2)) return reply('*The number must be a number!*')
          for (let i = 0; i < oi2; i++) {
          bosco.sendMessage(from, delb, MessageType.image, {caption: oi1})
          }
}
          break
      case 'delchat':
                    if (!isOwner && !mek.key.fromMe) return reply(mess.only.owner)
                reply('*Successfully delete chat*' + from)
                await sleep(4000)
                bosco.modifyChat(from, ChatModification.delete)
                break
     case 'mute':
                            if (!isOwner && !mek.key.fromMe) return reply(mess.only.owner)
                            if (!isGroup) return reply(mess.only.group)
                if (isMuted) return reply(`*already muted*`)
                mute.push(from)
                fs.writeFileSync('./database/mute.json', JSON.stringify(mute))
                reply('*The bot has been successfully muted in this chat*')
                break
     case 'tts':
                                        if (args.length < 1) return bosco.sendMessage(from, `Kode bahasanya
mana kak? contoh : ${prefix}tts id yamate kudasai`, text, { quoted: mek })
                                   const gtts = require('./lib/gtts')(args[0])
                                        if (args.length < 2) return bosco.sendMessage(from, `Teksnya mana
kak? contoh : ${prefix}tts id yamate kudasai`, text, { quoted: mek })
                                        var bby = body.slice(8)
                                        ranm = getRandom('.mp3')
                                        rano = getRandom('.ogg')
                                        bby.length > 300
                                                ? reply('Teks nya terlalu panjang kak')
                                                : gtts.save(ranm, bby, function () {
                                                        exec(`ffmpeg -i ${ranm} -ar 48000 -vn -c:a libopus
${rano}`, (err) => {
                                                                fs.unlinkSync(ranm)
                                                                buff = fs.readFileSync(rano)
                                                                if (err) return reply(dla.stikga())
                                                                bosco.sendMessage(from, buff, audio, {
duration: 359996400, ptt: true, quoted: mek })
                                                                fs.unlinkSync(rano)
                                                        })
                                                })
                                        break
        case 'demote':
                                if (!isGroup) return reply(mess.only.group)
                                if (!isGroupAdmins) return reply(mess.only.admin)
                                if (mek.message.extendedTextMessage === undefined ||
mek.message.extendedTextMessage === null) return reply('*Reply To Target*')
                        demote = mek.message.extendedTextMessage.contextInfo.participant
                    bosco.groupDemoteAdmin(from, [demote])
                                                reply('*Successful Demote an Admin*')
                                                break
                                        case 'promote':
                                        if (!isGroup) return reply(mess.only.group)
                                        if (!isGroupAdmins) return reply(mess.only.admin)
                                  if (mek.message.extendedTextMessage === undefined ||
mek.message.extendedTextMessage === null) return reply('*Reply To Target*')
                        promote = mek.message.extendedTextMessage.contextInfo.participant
                    bosco.groupMakeAdmin(from, [promote])
                                                reply('*Successful Promoted an Admin')
                                                break
                case 'linkgc':
                                if (!isGroup) return reply(mess.only.group)
                                        linkgc = await bosco.groupInviteCode(from)
                                        yeh = `https://chat.whatsapp.com/${linkgc}\n\nLink grup
${groupName}`
                                        bosco.sendMessage(from, yeh, text, { quoted: fgc })
                                        break
     case 'resetlinkgroup':
         case 'revoke':
         if (!isGroup) return reply(mess.only.group)
         if (!isGroupAdmins) return reply(mess.only.admin)
          json = ['action', 'inviteReset', from]
         bosco.query({json, expect200: true})
          reply('*Succes Reset Group Link*')
         break
     case 'tagme':
                  var nomqm = mek.participant
                                    tagu = `@${nomqm.split('@')[0]}`
                                        bosco.sendMessage(from, tagu, text, { quoted: ftrol, contextInfo: {
forwardingScore: 508, isForwarded: true, mentionedJid: [nomqm]}})
                                        break
                    case 'fdeface':
                              var nn = body.slice(9)
                                var urlnye = nn.split("|")[0];
                                var titlenye = nn.split("|")[1];
                                var descnye = nn.split("|")[2];
                                imgbbb = require('imgbb-uploader')
                                run = getRandom('.jpeg')
                                encmediad = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                                mediad = await bosco.downloadAndSaveMediaMessage(encmediad)
                                ddatae = await imageToBase64(JSON.stringify(mediad).replace(/\"/gi, ''))
                                bosco.sendMessage(from, {
                                        text: `${urlnye}`,
                                        matchedText: `${urlnye}`,
                                        canonicalUrl: `${urlnye}`,
                                        description: `${descnye}`,
                                        title: `${titlenye}`,
                                        jpegThumbnail: ddatae
                                }, 'extendedTextMessage', { detectLinks: false })
                                break

                case 'fitnahpc':
                if (args.length < 1) return reply(`Usage :\n${prefix}fitnahpc
[number|message|replybot]]\n\nEx : \n${prefix}fitnahpc 0|hai|hai juga markenlin`)
                var gh = body.slice(10)
                var parti = gh.split("|")[0];
                var targetq = gh.split("|")[1];
                                var bot = gh.split("|")[2];
                              bosco.sendMessage(from, `${bot}`, text, {quoted: { key: { fromMe: false,
participant: `${parti}@s.whatsapp.net`, ...(from ? { remoteJid: "status@broadcast" } : {}) }, message: {
conversation: `${targetq}` }}})
                                        break
                case 'return':
                                if (!isOwner) return
                                        return bosco.sendMessage(from, JSON.stringify(eval(body.slice(8))),
text, {quoted: mek})
                                        if (err) return bosco.sendMessage(from, `root @Denis Ser:~ ${err}`,
text, { quoted: mek })
                 break
              case 'swm':
                                                if (isMedia && !mek.message.videoMessage || isQuotedImage)
{
                                                        ppp = `${args.join(' ')}`
                                                        const encmediao = isQuotedImage ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                                                        const media = await
bosco.downloadAndSaveMediaMessage(encmediao, `./sticker/${sender}`)
                                                        const packname1 = ppp.split('|')[0]
                                                        const author1 = ppp.split('|')[1]
                                                        exif.create(packname1, author1,
`stickwm_${sender}`)
                                                        await ffmpeg(`${media}`)
```

```
                                                                                .input(media)
                                                                                .on('start', function (cmd) {
                                                                                        console.log(`Started :
${cmd}`)
                                                                                })
                                                                                .on('error', function (err) {
                                                                                        console.log(`Error :
${err}`)

                                                                                        fs.unlinkSync(media)
                                                                                        reply(mess.error.api)
                                                                                })
                                                                                .on('end', function () {
                                                                                        console.log('Finish')
                                                                                        exec(`webpmux -set exif
./sticker/stickwm_${sender}.exif ./sticker/${sender}.webp -o ./sticker/${sender}.webp`, async (error) => {
                                                                                                if (error) return
reply(mess.error.api)

bosco.sendMessage(from, fs.readFileSync(`./sticker/${sender}.webp`), sticker, {quoted: mek})

fs.unlinkSync(media)

fs.unlinkSync(`./sticker/${sender}.webp`)

fs.unlinkSync(`./sticker/stickwm_${sender}.exif`)
                                                                                        })
                                                                                })
                                                                                .addOutputOptions([`-
vcodec`,`libwebp`,`-vf`,`scale='min(320,iw)':min'(320,ih)':force_original_aspect_ratio=decrease,fps=15,
pad=320:320:-1:-1:color=white@0.0, split [a][b]; [a]
palettegen=reserve_transparent=on:transparency_color=ffffff [p]; [b][p] paletteuse`])
                                                                                .toFormat('webp')
                                                                                .save(`./sticker/${sender}.webp`)
                                                } else if ((isMedia && mek.message.videoMessage.fileLength
< 10000000 || isQuotedVideo &&
mek.message.extendedTextMessage.contextInfo.quotedMessage.videoMessage.fileLength < 10000000)) {
                                                        wmsti = body.slice(11)
                                                        if (!wmsti.includes('|')) return reply(`Kirim
gambar atau reply gambar dengan caption *${prefix}stickerwm nama|author*`)
                                                        const encmedia = isQuotedVideo ?
JSON.parse(JSON.stringify(mek).replace('quotedM', 'm')).message.extendedTextMessage.contextInfo : mek
                                                        const media = await
bosco.downloadAndSaveMediaMessage(encmedia, `./sticker/${sender}`)
                                                        const packname1 = wmsti.split('|')[0]
                                                        const author1 = wmsti.split('|')[1]
                                                        exif.create(packname1, author1,
`stickwm_${sender}`)
                                                        reply(mess.wait)
                                                                await ffmpeg(`${media}`)
                                                                                .inputFormat(media.split('.')[4])
                                                                                .on('start', function (cmd) {
                                                                                        console.log(`Started :
${cmd}`)
                                                                                })
                                                                                .on('error', function (err) {
                                                                                        console.log(`Error :
${err}`)

                                                                                        fs.unlinkSync(media)
                                                                                        tipe =
media.endsWith('.mp4') ? 'video' : 'gif'

                                                                                        reply(mess.error.api)
                                                                                })
                                                                                .on('end', function () {
                                                                                        console.log('Finish')
                                                                                        exec(`webpmux -set exif
./sticker/stickwm_${sender}.exif ./sticker/${sender}.webp -o ./sticker/${sender}.webp`, async (error) => {
                                                                                                if (error) return
reply(mess.error.api)

bosco.sendMessage(from, fs.readFileSync(`./sticker/${sender}.webp`), sticker, {quoted: mek})

fs.unlinkSync(media)

fs.unlinkSync(`./sticker/${sender}.webp`)

fs.unlinkSync(`./sticker/stickwm_${sender}.exif`)
                                                                                        })
                                                                                })
                                                                                .addOutputOptions([`-
vcodec`,`libwebp`,`-vf`,`scale='min(320,iw)':min'(320,ih)':force_original_aspect_ratio=decrease,fps=15,
pad=320:320:-1:-1:color=white@0.0, split [a][b]; [a]
palettegen=reserve_transparent=on:transparency_color=ffffff [p]; [b][p] paletteuse`])
                                                                                .toFormat('webp')
```

````
                                                                        .save(`./sticker/${sender}.webp`)
                                                } else {
                                                        reply(`*Send picture/video with caption
${prefix}stickerwm nama|author or tag images/videos that have been sent*\n*Note: Maximum video duration is
10 seconds*`)
                                                }
                                                break









default:
if (budy.startsWith('=>')){
if (!isOwner) return
try {
return bosco.sendMessage(from,
`${a}📥 Input: ${budy.slice(3)}
📤 OutPut:
${JSON.stringify(eval(budy.slice(2)),null,'\t')}
${a}`
,text, {quoted:mek })
} catch(err) {
e = String(err)
reply(`${a} "err" :  "${e}"${a}`)
}
}
if (budy.startsWith(`$`)){
if (!isOwner) return
const sep = budy.split("\n")
let exc = budy.replace(sep[0]+"\n", "")
exec(exc, (err, stdout) => {
if (err) return bosco.sendMessage(from, `root @denis:~ ${err}`, text, { quoted: mek })
if (stdout) {
bosco.sendMessage(from, stdout, text)
}
})
}
if (budy.startsWith('>')) {
if (!isOwner) return
try {
console.log(color('[ EVAL ]', 'pink'), color(time), budy, color('from', 'yellow'), pushname, color('di'),
isGroup ? groupName : 'Private Chat')
reply(require('util').format(eval(`;(async () => { ${budy.slice(2)} })()`)))
} catch(e) {
console.log(e)
err = String(e)
js = JSON.stringify(e, null, 2)
if (js == '{}') js = { err }
js = JSON.stringify(js, null, 2)
js = '```' + js + '```'
reply('_' + err + '_\n\n' + js)
}
}
if (isGroup && budy != undefined) {
} else {
console.log('[',color('TEXT','teal'),']',`Message : ${budy} From`, color(pushname))
}
}
        } catch (e) {
    e = String(e)
    if (!e.includes("this.isZero")) {
        console.log('Message : %s', color(e, 'green'))
        }
        }
}
````