**完成创建自定义机器人并且获取自定义机器人Webhook 地址：**

<https://oapi.dingtalk.com/robot/send?access_token=cf57afa18dee4208ea4b37fe285bba2beef54d077b3ac7aa1ed46e5cb807d387>

**快速验证自定义机器人是否可以正常工作：**

curl 'https://oapi.dingtalk.com/robot/send?access\_token=cf57afa18dee4208ea4b37fe285bba2beef54d077b3ac7aa1ed46e5cb807d387' \

-H 'Content-Type: application/json' \

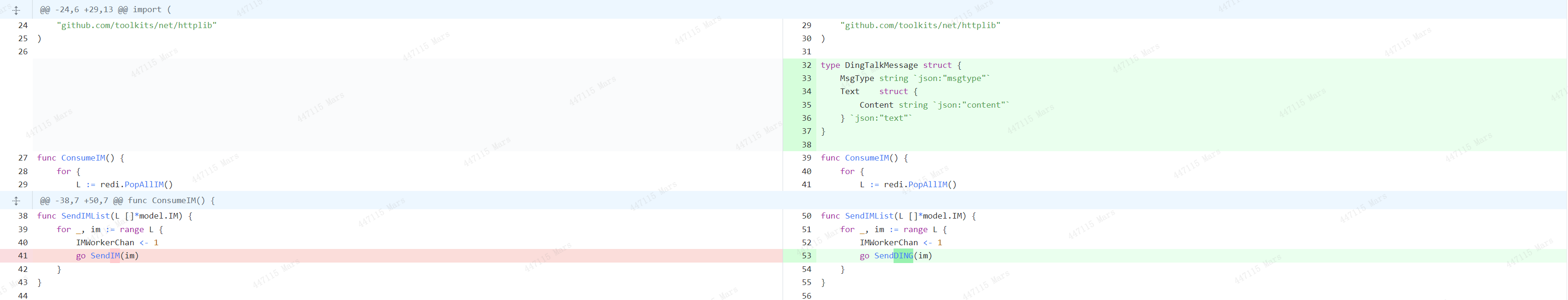
-d '{"msgtype": "text","text": {"content":"监控报警测试"}}'

-d ‘{"at": "atAll":true}’

**主要代码修改：**

由于机器人Webhook 地址较长，需要修改数据库的im字段长度，这里参考设置为4000。已在远程数据库上进行修改。

在modules\alarm\cron\im\_sender.go中对代码进行修改，加入钉钉机器人消息的结构，并且修改原调用im处理的函数为调用钉钉机器人处理的函数

****

添加钉钉机器人发送函数SendDING(im \*model.IM)，用im字段存放钉钉机器人的Webhook 地址，通过Webhook 地址让钉钉机器人发送报警信息。

func SendDING(im \*model.IM) {

    defer func() {

        <-IMWorkerChan

    }()

    dingTalkMessage := DingTalkMessage{

        MsgType: "text",

        Text: struct {

            Content string `json:"content"`

        }{

            Content: im.Content,

        },

    }

    // 序列化消息体为JSON

    jsonValue, err := json.Marshal(dingTalkMessage)

    if err != nil {

        log.Errorf("序列化消息体为JSON fail, dingTalkMessage:%s, error:%v", dingTalkMessage, err)

    }

    // 发送HTTP POST请求

    resp, err := http.Post(im.Tos, "application/json", bytes.NewBuffer(jsonValue))

    if err != nil {

        log.Errorf("send dingding fail, tos:%s, content:%s, dingTalkMessage:%s, error:%v,", im.Tos, im.Content, dingTalkMessage, err)

    }

    defer resp.Body.Close()

    // 读取响应体

    body, \_ := ioutil.ReadAll(resp.Body)

    fmt.Printf("Response: %s\n", body)

    // 检查响应状态码

    if resp.StatusCode != http.StatusOK {

        fmt.Printf("HTTP Status: %s, Body: %s", resp.Status, body)

    }

    log.Debugf("send im:%v, resp:%v, url:%s", im, resp, im.Tos)

}

**使用方法：**

需要新加钉钉群进行通知时，在dashboard界面中新建一个用户，用户的im字段填写为新加钉钉群中机器人的Webhook 地址，即可通过机器人发送报警通知。

