SSH任务总结

一个测试Demo：

Demo背景：前端主动像后端发起websocket请求并且携带相关数据，后端先进性判断，如果验证成功就可以放行进行websocket连接，此时前后端可以畅通无阻的通话。

Demo代码：

```go
package main

import (
    "fmt"
    "github.com/gorilla/websocket"
    "net/http"
)

var upgrader = websocket.Upgrader{
    CheckOrigin: func(r *http.Request) bool {
        return true // 允许所有CORS请求
    },
}

func serveWs(w http.ResponseWriter, r *http.Request) {
    // 从查询参数获取验证信息
    username := r.URL.Query().Get("username")
    password := r.URL.Query().Get("password")

    // 验证逻辑
    if username != "user" || password != "pass" {
        http.Error(w, "错误请求", http.StatusUnauthorized)
        return
    }

    // 升级HTTP连接到WebSocket连接
    ws, err := upgrader.Upgrade(w, r, nil)
    if err != nil {
        fmt.Println("Error upgrading to WebSocket:", err)
        return
    }
    defer ws.Close()

    for {
        var msg map[string]string
        // 读取消息
        err := ws.ReadJSON(&msg)
        if err != nil {
            fmt.Println("Error reading json:", err)
            break
        }

        // 打印并回复消息
        fmt.Printf("Received message: %s\n", msg["data"])
```

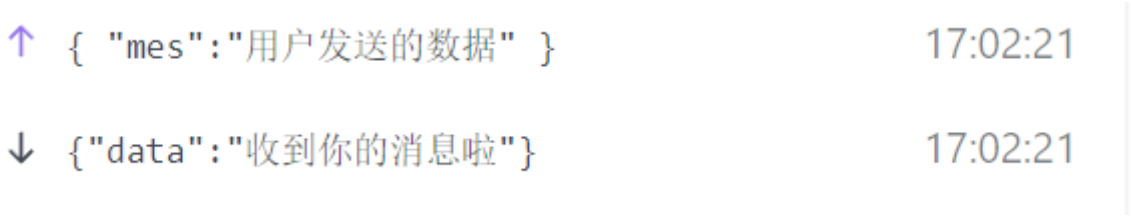
```
        err = ws.WriteJSON(map[string]string{"data": "收到你的消息啦"+
msg["data"]}})
        if err != nil {
            fmt.Println("Error writing json:", err)
            break
        }
    }
}

func main() {
    http.HandleFunc("/ws", serveWs)
    fmt.Println("Server started on :8080")
    err := http.ListenAndServe(":8080", nil)
    if err != nil {
        fmt.Println("ListenAndServe: ", err)
    }
}
```

注意这个Demo的参数在url上。

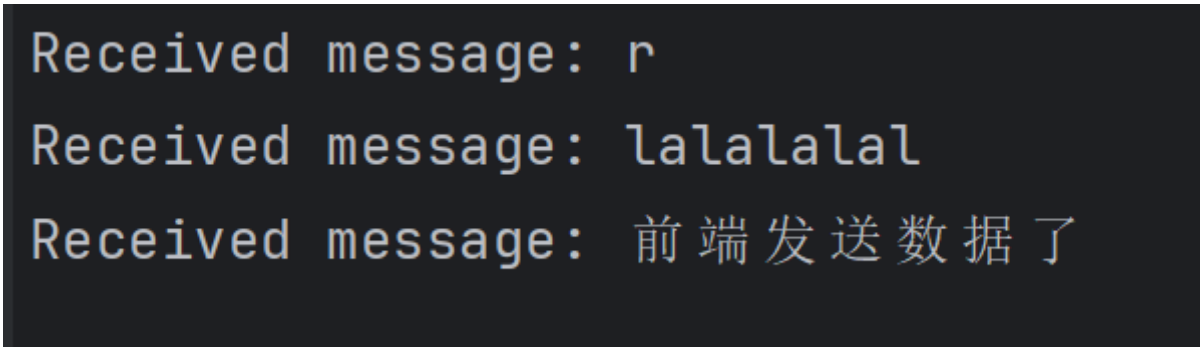
下图是用ApiFox建立连接

连接建立完成后，在message中输入信息（下图）

此时，后端会自动返回数据。上箭头代表用户（发送数据），下箭头是后端（接受后端数据）

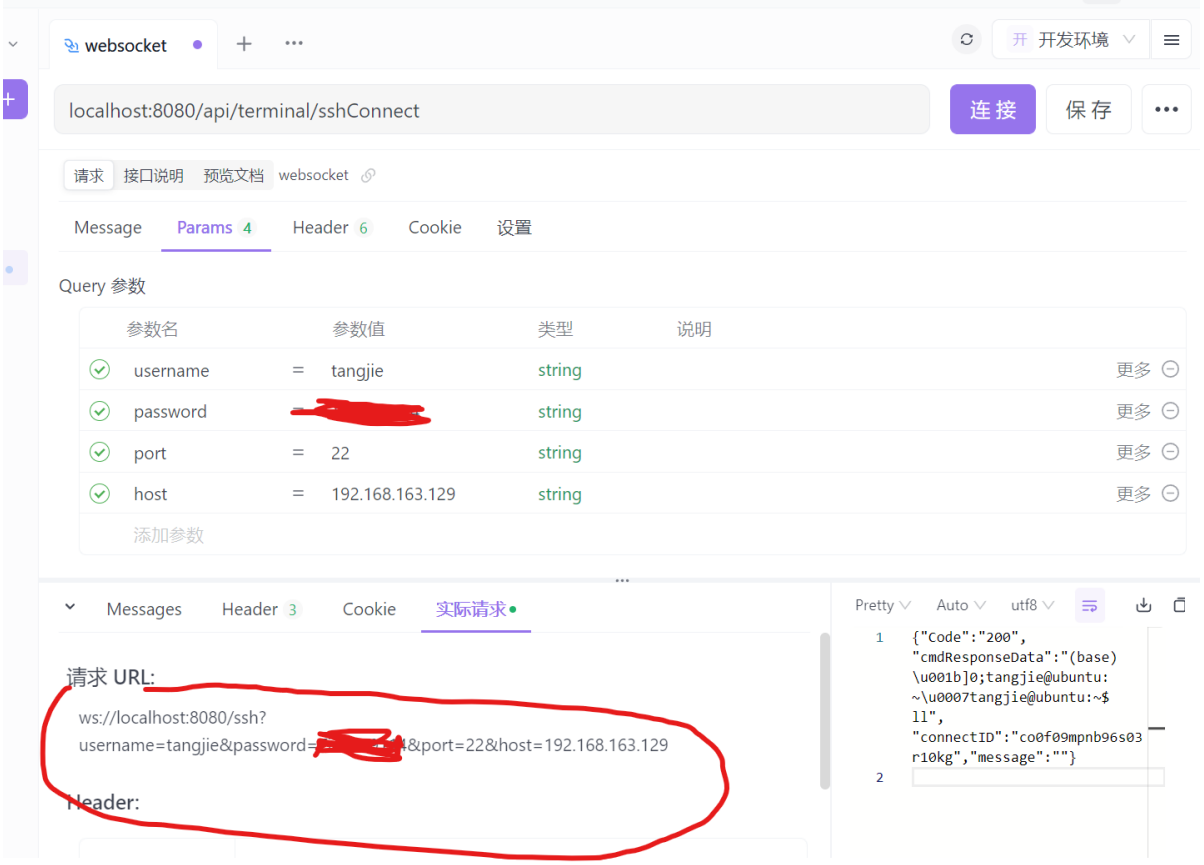


在后端可以收到数据：



关于远程连接

请求：



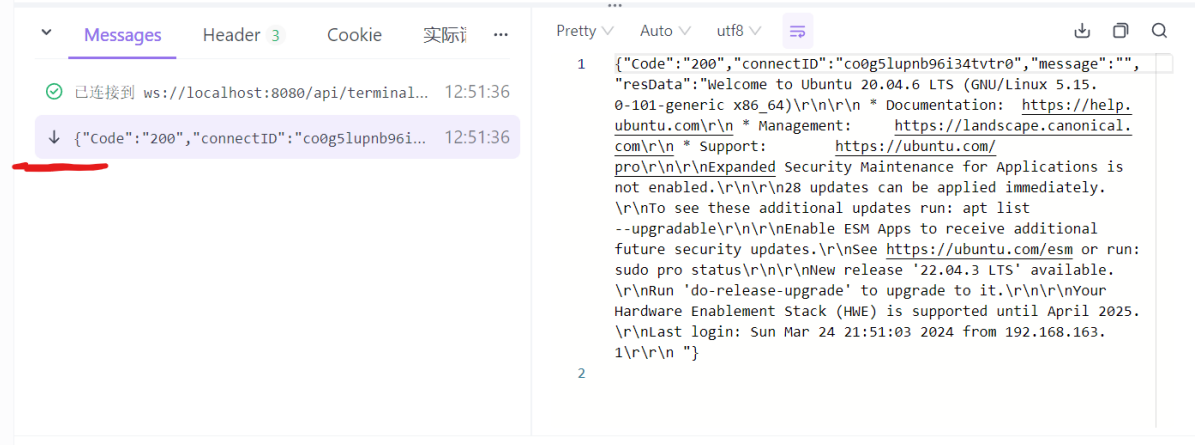
在ApiFox中有ws关键字用于websocket请求。

在后端代码中，我会根据你传递的虚拟机信息先进行连接测试（看看能否正确连接连接虚拟机）

```go
func sshConnect(w http.ResponseWriter, r *http.Request) {  1 usage

    host := r.URL.Query().Get( key: "host")
    port := r.URL.Query().Get( key: "port")
    password := r.URL.Query().Get( key: "password")
    username := r.URL.Query().Get( key: "username")

    vmInfo := model.VMConnectRequest{
        Host:     host,
        Port:     port,
        Username: username,
        Password: password,
    }

    config := &ssh.ClientConfig{
        User: vmInfo.Username,
        Auth: []ssh.AuthMethod{
            ssh.Password(vmInfo.Password),
        },
        HostKeyCallback: ssh.InsecureIgnoreHostKey(),
    }

    //测试连接
    flag, connctID := utils.VerifyConnect(&vmInfo)
    if flag != true {
        //todo 返回错误码
        log.Fatalln( v...: "001测试连接失败")
        return
    }
```

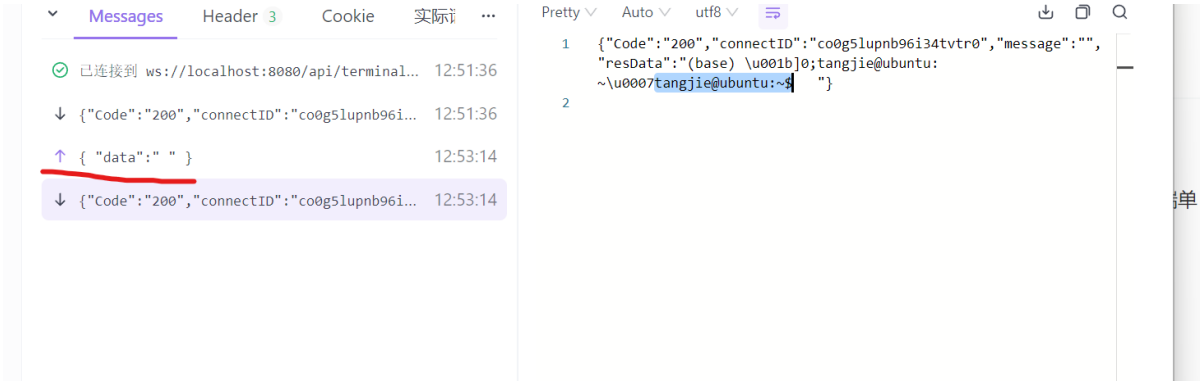
如果不能就直接退出程序，如果可以连接就会响应成功（code： 200）

要注意，在建立完websocket连接后，后端会将终端的初始信息 发送到websocket中



上图 只有一个下箭头表示这个数据是自动发送的。

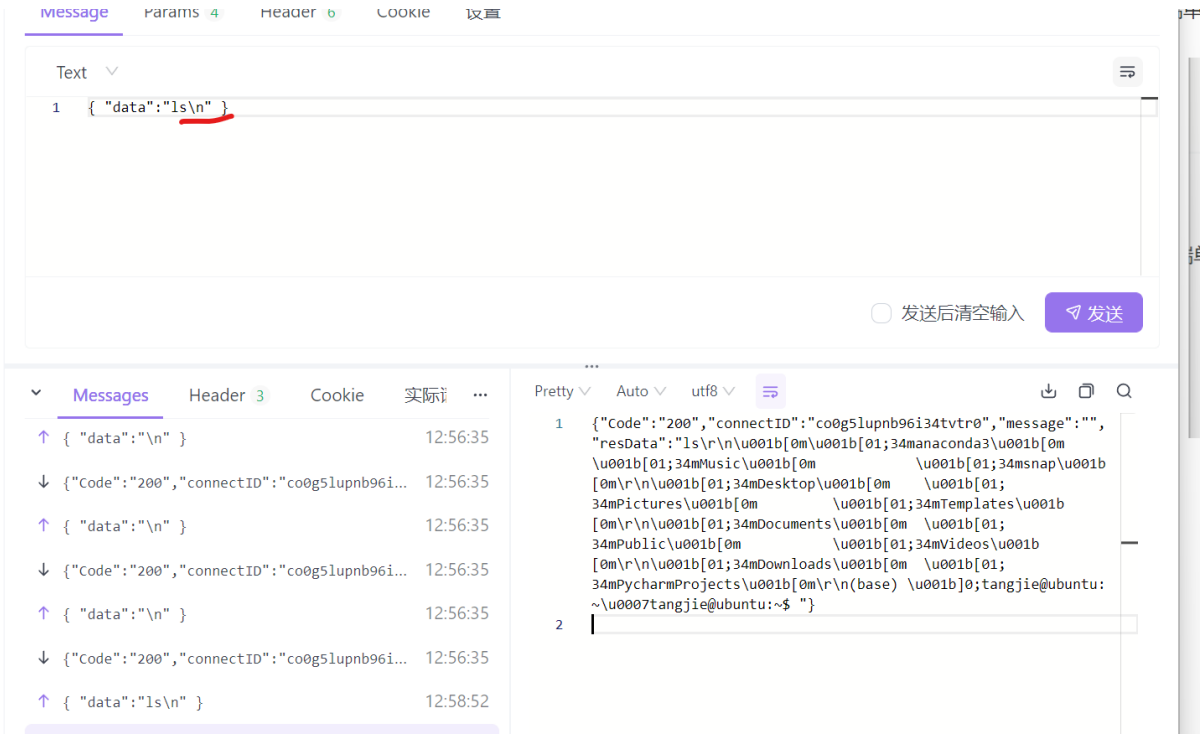
但是注意这个信息中没有用户的提示符 tangjie@ubuntu:~$ ，这个符号并没有被虚拟机作为初始化信息在第一条数据发送，需要前端单独发一条数据才能显示（上箭头是前端）



所以前端需要在接收第一条初始化信息后再发送一条数据（空格即可，注意不要发送空字符串"", 这样会解析错误）

同时虚拟机的返回有许多特殊字符，需要前端进行解析 。

下图展示了发送命令



在这个测试命令中 用\n表示enter键，右侧是响应数据。

另外要注意，如果要实现补全功能（tab键）可能前端要进行额外处理，ios手机没有tab键。这个问题后端无法解决，只能依赖前端