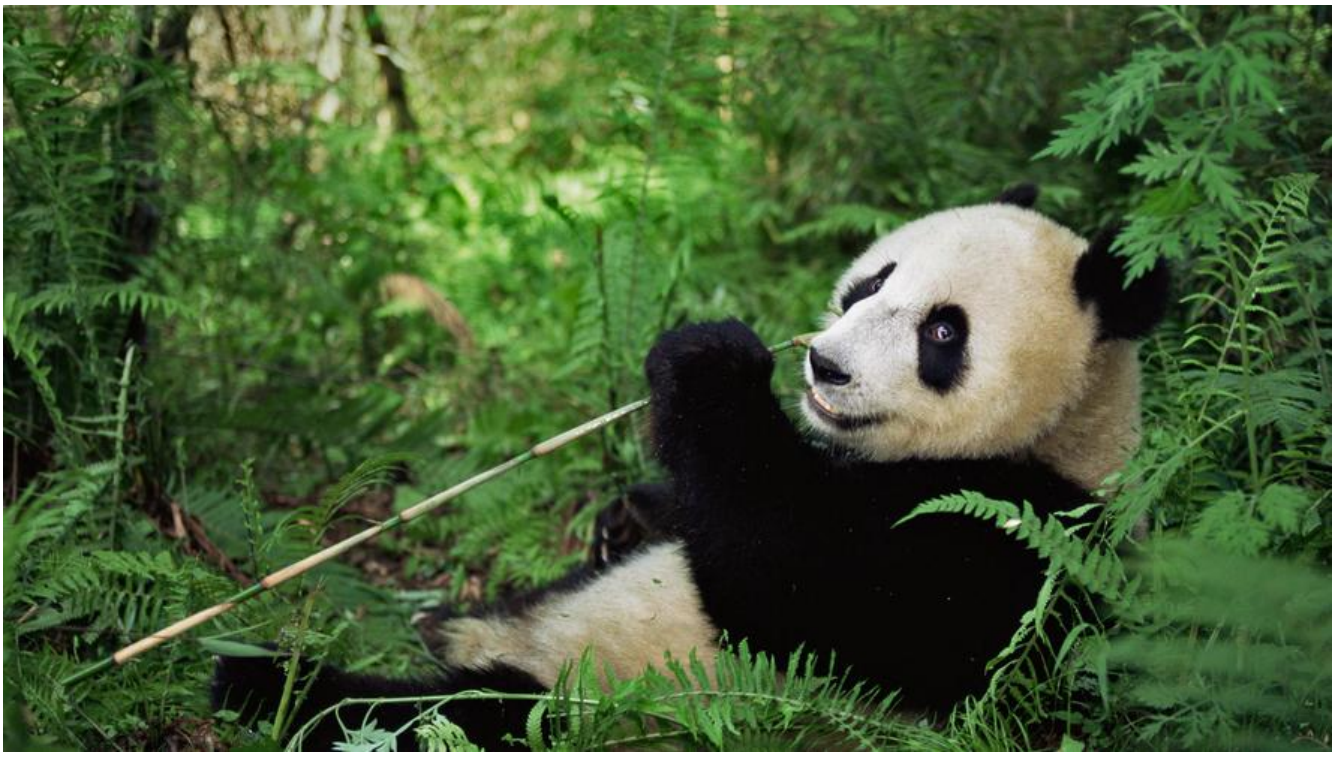
链滴

【GO 学习笔记】goland 调试 docker 容器中的 go 代码

作者：Archer1A

原文链接：https://ld246.com/article/1575700409784

来源网站：链滴



goland 调试docker容器中的go代码

前言:为什么需要去远程容器中的go代码?

最近正在看Harbor的源码，由于本人电脑是windows 无法直接启动进行debug,后来搭了一个CentOS的桌面环境,又发现启动需要很多很多环境参数...而且还有很多其他组件都是运行在docker上,使用linu系统启动的方案又失败了！后来查了好多资料,找到delve这个插件,该插件可以在docker容器中，远程试代码。这里记录下过程(使用的是小demo而不是Harbor)！

一、 下载安装delve

1. 进入 GOPATH路径创建对应目录

```
mkdir -p $GOPATH/github.com/go-delve
cd $GOPATH/github.com/go-delve
```

2. github clone delve 源码

```
$ git clone https://github.com/go-delve/delve.git
```

3. 编译

```
$ cd $GOPATH/src
$ go install github.com/go-delve/delve/cmd/dlv
```

4. 编译完成后dlv文件就在 $GOPATH/bin目录下了，可以以下命令查看版本信息：

```
$ dlv -v
```

二、 准备Demo

1. 准备一个demo,文件名为web.go

```
package main

import (
    "crypto/md5"
    "fmt"
    "html/template"
    "io"
    "log"
    "net/http"
    "os"
    "strconv"
    "strings"
    "time"
)

func sayHelloName(w http.ResponseWriter,r *http.Request)  {
    r.ParseForm() // 解析参数，默认是不会解析的
    fmt.Println(r.Form) // 这些信息是输出入服务器的打印信息
    fmt.Println("path " ,r.URL.Path)
    fmt.Println("scheme " , r.URL.Scheme)
    fmt.Println(r.Form["url_long"])
    for k,v := range r.Form{
        fmt.Println("key:" ,k)
        fmt.Println("value",strings.Join(v,""))
    }
    fmt.Fprintf(w,"hello vic")

}

func login(w http.ResponseWriter,r *http.Request)  {
    fmt.Println("method " ,r.Method)
    if r.Method == "GET" {
        t,_ := template.ParseFiles("login.html")
        log.Println(t.Execute(w, nil))
    } else{
        r.ParseForm() // 如果注释这句不会有输出 默认情况下，Handler里面是不会自动解析form的
必须显式的调用r.ParseForm()后，你才能对这个表单数据进行操作
        //请求的是登录数据，那么执行登录的逻辑判断
        fmt.Println("username:", r.Form["username"])
        fmt.Println("password:", r.Form["password"])
    }



}
func upload(w http.ResponseWriter,r *http.Request)  {
    fmt.Println("method:", r.Method) //获取请求的方法
    if r.Method == "GET" {
        crutime := time.Now().Unix()
        h := md5.New()
```

```
        io.WriteString(h,strconv.FormatInt(crutime,10))
        token := fmt.Sprintf("/x",h.Sum(nil))

        t,_ := template.ParseFiles("upload.html")
        fmt.Println(token)
        t.Execute(w,nil)

    } else{
        r.ParseMultipartForm(32 << 20)
        file,handler,err := r.FormFile("uploadfile")
        if err != nil {
            fmt.Println(err)
            return
        }
        defer file.Close()
        fmt.Fprintf(w, "%v", handler.Header)
        f, err := os.OpenFile("./test/"+handler.Filename, os.O_WRONLY|os.O_CREATE, 0666)  //
处假设当前目录下已存在test目录
        if err != nil {
            fmt.Println(err)
            return
        }
        defer f.Close()
        io.Copy(f, file)

    }

}

func main()  {
    http.HandleFunc("/v2",sayHelloName)//设置访问路由
    http.HandleFunc("/login", login)
    http.HandleFunc("/upload", upload)
    err := http.ListenAndServe(":9090",nil)// 设置监听端口
    if err != nil {
        log.Fatal("ListenAndServe: " ,err)
    }

}
```

三、制作镜像

1. 准备启动脚本start.sh

```
#!/bin/bash
dlv debug --headless --listen=:2345 --api-version=2 --accept-multiclient   $GOPATH/src/web
go
```

2. 创建Dockerfile,并将以上所有文件放在同一个目录中，目录下有4个文件

```
[root@localhost go-debug]# ll
总用量 17708
```

```
-rwxr-xr-x. 1 root root 18119408 12月  3 14:52 dlv
-rw-r--r--. 1 root root      187 12月  3 18:55 Dockerfile
-rw-r--r--. 1 root root      107 12月  3 16:21 start.sh
-rw-r--r--. 1 root root     2168 12月  3 14:49 web.go
```

3. 编写Dockerfile

```
FROM golang:1.12.12

RUN mkdir /web/

COPY ./dlv /usr/local/bin/
COPY ./web.go /go/src
COPY ./start.sh  /web/
RUN chmod   u+x /web/start.sh
WORKDIR /go/src/
ENTRYPOINT ["/web/start.sh"]
```

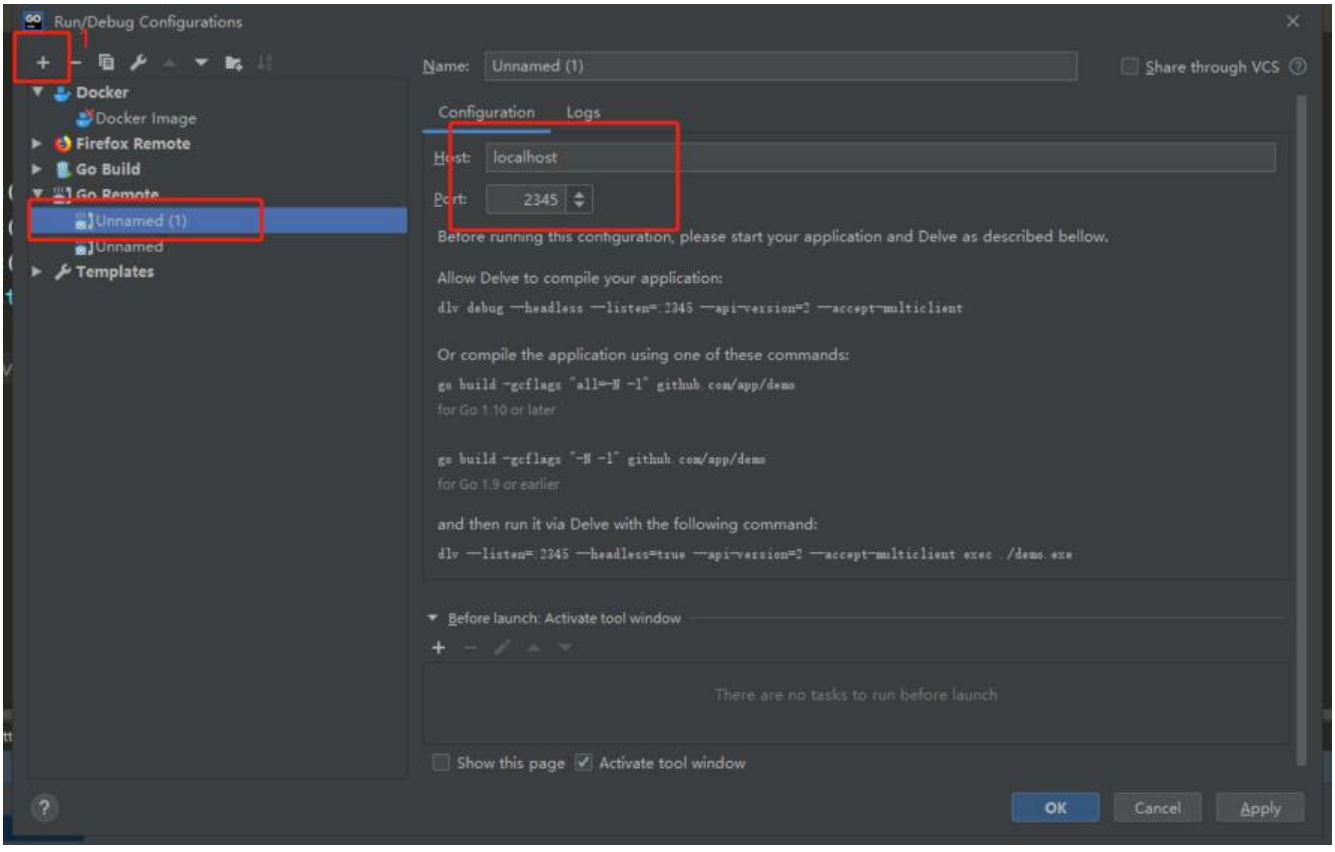
4. 构建镜像

```
$ docker build . -t web:1.0
```

5. 运行镜像

```
$ docker run -d  -p 9090:9090 -p 2345:2345 --privileged  web:1.0
```

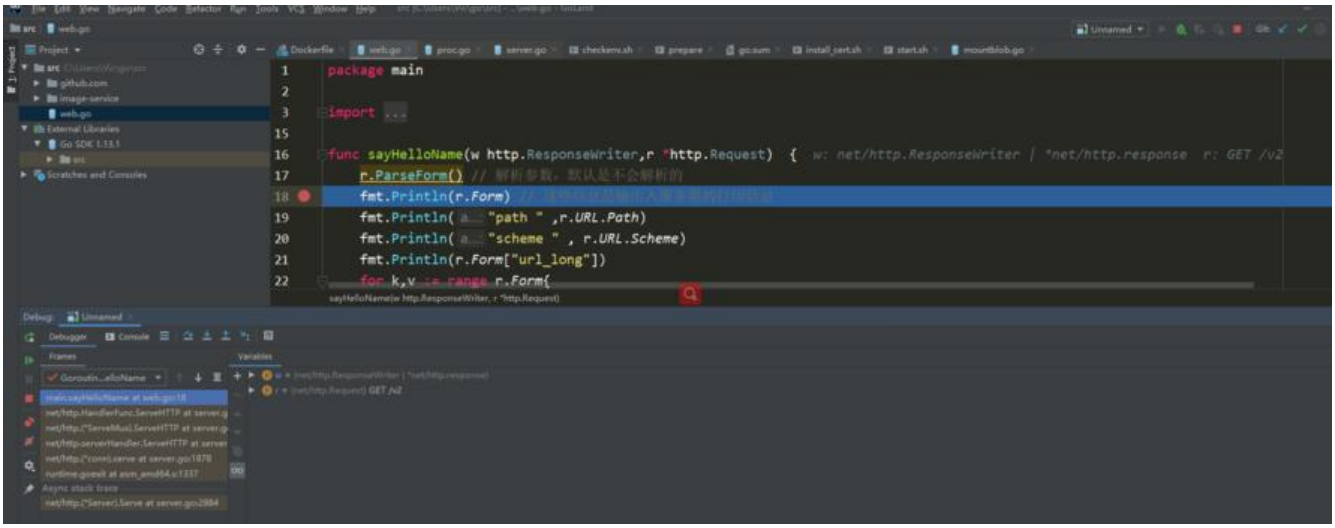
四、goland的配置



这里有个坑就是，远程被调试的代码和本地代码都必须在$GOAPTH/src目录下！！！

goland 单击debug 按钮 打上断点 接下来就可以远程调试了！！访问接口，就发现代码停在了断点了

$ curl localhost:9090/v2



整个流程结束！

源码地址：

- github

五、后续思路

1. 修改启动脚本,代码通过挂载的方式加到容器中，应该可以达成不用重复制作镜像的效果
2. 用harbor源码调试，这样就可以进行愉快的debug了 而不是光靠想象了