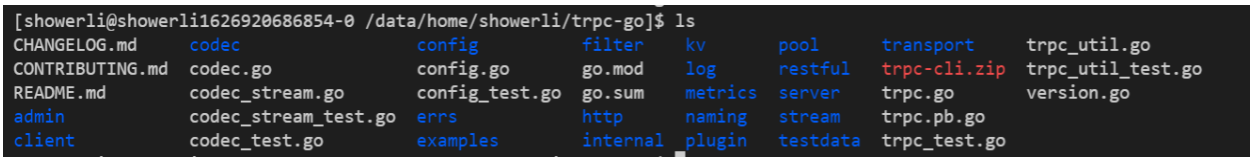
环境搭建

1. tRPC-Go容器

参考[tRPC-Go环境搭建 - 腾讯iWiki (woa.com)](), 申请云容器并登录容器配置，将trpc-go配置在showerli账号下：



2. 安装相应工具

trpc和trpc-cli，参考[trpc-cli 快速上手 - 腾讯iWiki (woa.com)]()

项目实现

1. 定义服务接口

tRPC采用protobuf来描述一个服务，我们用protobuf定义服务方法，请求参数和响应参数，cd到前面创建好的kv目录里面并编辑pb文件

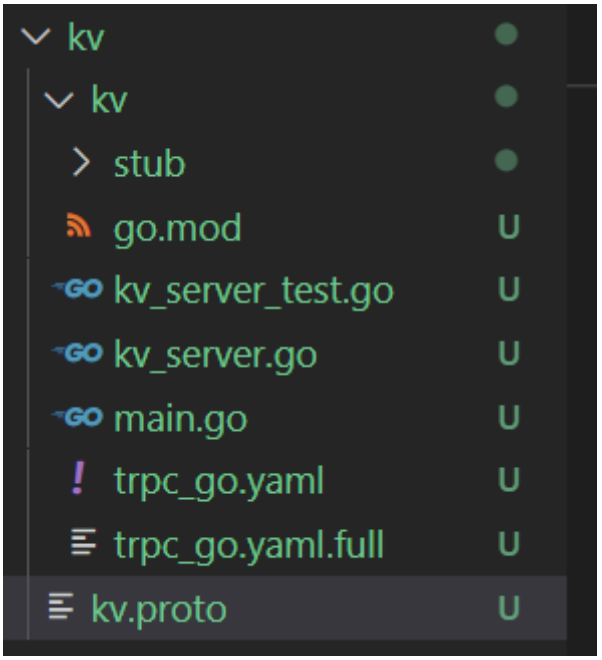
```
// syntax必须是proto3，tRPC都是基于proto3通信的。
syntax = "proto3";

// package内容格式推荐为trpc.{app}.{server}，以trpc为固定前缀，标识这是一个trpc服
package trpc.kv;

// 这里没有为服务单独创建一个git，只有一个本地目录kv
option go_package="kv";

// 定义服务接口
service KVServer {
rpc Get (getRequest) returns (valueReply) {}
rpc Set (setRequest) returns (valueReply) {}
rpc BeginTx (getRequest) returns (valueReply) {}
rpc Rollback (getRequest) returns (valueReply) {}
rpc Commit (getRequest) returns (valueReply) {}
```

```
rpc Prefix (getRequest) returns (valueReply) {}
}

// 请求参数
message getRequest {
string key = 1;
}

// 请求参数
message setRequest {
string key = 1;
string value = 2;
}

// 响应参数
message valueReply {
string msg = 1;
}
```

2. 生成服务代码

```
[root@showerli1626920686854-0 /data/home/showerli/trpc-go/kv]# trpc create --protofile=kv.proto
[Info] Found protofile:kv.proto in following dir:/data/home/showerli/trpc-go/kv
[Info] Generate project ```kv.proto``` success
```

然后可以看到目录里面生成了很多文件：



trpc_go.yaml： 是整个项目的配置文件，里面可以更改监听的ip地址和端口号，这里保持默认配置

main.go: main()定义，这里也不需要修改

main.go U

kv > kv > main.go

```go
package main

import (
    _ "git.code.oa.com/trpc-go/trpc-config-tconf"
    _ "git.code.oa.com/trpc-go/trpc-filter/debuglog"
    _ "git.code.oa.com/trpc-go/trpc-filter/recovery"
    _ "git.code.oa.com/trpc-go/trpc-log-atta"
    _ "git.code.oa.com/trpc-go/trpc-metrics-m007"
    _ "git.code.oa.com/trpc-go/trpc-metrics-runtime"
    _ "git.code.oa.com/trpc-go/trpc-naming-polaris"
    _ "git.code.oa.com/trpc-go/trpc-opentracing-tjg"
    _ "git.code.oa.com/trpc-go/trpc-selector-cl5"

    trpc "git.code.oa.com/trpc-go/trpc-go"
    pb "kv"

    _ "go.uber.org/automaxprocs"
)

type kVServerServiceImpl struct{}

func main() {

    // 创建一个服务对象，底层会自动读取服务配置及初始化插件，必须放在main函数首行，业务初始化逻辑必须放在NewServer后面
    s := trpc.NewServer()

    // 注册当前实现到服务对象中
    pb.RegisterKVServerService(s, &kVServerServiceImpl{})

    // 启动服务，并阻塞在这里
    if err := s.Serve(); err != nil {
        panic(err)
    }
}
```

3. 开发具体业务逻辑

打开kv_server.go并编辑：

```go
package main

import (
    "context"
    "io/ioutil"
    "encoding/json"
    "strings"

    pb "kv"
)

```

```go
var filename = "kv_data"
var ifBegin = false
var tempMap = make(map[string]string) // 利用临时map来储存


// Get方法
func (s *kVServerServiceImpl) Get(ctx context.Context, req *pb.GetRequest, r
    // 打开json格式的文件
    jsonFile, err := ioutil.ReadFile(filename);
    if err != nil {
        rsp.Msg = "open file failed"
        }

    kv_map := make(map[string]string)

    if err := json.Unmarshal(jsonFile, &kv_map) ; err != nil {
        rsp.Msg = "open json file failed"
    } else {
        key := req.Key
        value, ok := kv_map[key]
        if ok {
            rsp.Msg = value
        } else{
            rsp.Msg = "not found in kv_data"
        }
    }
    return nil
}

// Set方法
func (s *kVServerServiceImpl) Set(ctx context.Context, req *pb.SetRequest, r
   if ! ifBegin {  // 没有提交
       rsp.Msg = "please begin first"
   } else {
       tempMap[req.Key] = req.Value
       rsp.Msg = "set successfully"
   }
    return nil
}

// BeginTx
func (s *kVServerServiceImpl) BeginTx(ctx context.Context, req *pb.GetReques
    ifBegin = true
```

```
    rsp.Msg = " ********begin******** "
    return nil
}

// Rollback
func (s *kVServerServiceImpl) Rollback(ctx context.Context, req *pb.GetReque
    tempMap = make(map[string]string)
    ifBegin = false
    rsp.Msg = " ********rollback success******** "
    return nil
}

// Commit
func (s *kVServerServiceImpl) Commit(ctx context.Context, req *pb.GetRequest
    if len(tempMap) == 0 {
        rsp.Msg = "no change found"
    } else{
        // MarshalIndent应用Indent格式化输出,输出中的每个JSON元素均将从换行符开头
        // 该行以前缀开头，然后根据缩进嵌套嵌套一个或多个缩进副本
        if kv_data, err := json.MarshalIndent(tempMap,"", " ") ;err != nil {
            rsp.Msg = "json file failed"
        } else{
            // /将kv数据写入文件中，并修改文件权限为755
            if err = ioutil.WriteFile(filename, kv_data, 0755) ;err != nil {
                rsp.Msg = "write failed"
            } else {
            tempMap = make(map[string]string)
            ifBegin = false
            rsp.Msg = " ********commit success******** "
            }
        }
    }
    return nil
}

// Prefix
func (s *kVServerServiceImpl) Prefix(ctx context.Context, req *pb.GetRequest
    file, err := ioutil.ReadFile(filename)
    if err!=nil {
    rsp.Msg = "open file failed!"
}

kv_map :=make(map[string]string)
```

```
98
99  if err := json.Unmarshal(file, &kv_map) ; err !=nil{
100     rsp.Msg = "error: json file failed!"
101     } else {
102         for key, value := range kv_map {
103             if strings.HasPrefix(key, req.Key){
104             rsp.Msg += value
105             rsp.Msg += " , "
106         }
107     }
108 }
109     return nil
110 }
111
```

4. 开始编译，根目录执行go build

这里有个坑，可能会碰到permission denied报错：

```
[showerli@showerli1626920686854-0 /data/home/showerli/trpc-go/kv/kv]$ go build
error writing go.mod: open /data/home/showerli/trpc-go/kv/kv/go.mod298498081.tmp:
permission denied
```

于是切换到root账户，修改权限即可：

```
1 [root@showerli1626920686854-0 ~]# chmod -R 777 /data/home/showerli/trpc-go/kv
```

5. 启动服务

```
[showerli@showerli1626920686854-0 /data/home/showerli/trpc-go/kv/kv]$ ./trpc.app.K
VServer
2021/07/26 15:59:27 maxprocs: Updating GOMAXPROCS=4: determined from CPU quota
2021-07-26 15:59:27.377 DEBUG   maxprocs/maxprocs.go:47 maxprocs: Updating GOMAXPR
OCS=4: determined from CPU quota
2021-07-26 15:59:27.377 INFO    server/service.go:134   process:550247, trpc servi
ce:trpc.kv.KVServer launch success, tcp:127.0.0.1:8000, serving ...
```

6. 自测

设置值，会提示先开启事务：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Set -targe
t ip://127.0.0.1:8000 -body '{"key":"name", "value":"showerli"}'

[rsp json body]:{"msg":"please begin first"}
```

func为pb协议定义的 /package.service/method，在**trpc_go.yaml**里面可以找到。

target为被调服务的目标地址，格式为selectorname://servicename，这里只是本地自测，没有接入名字服务，直接指定ipport寻址，使用ip selector就可以了，格式是ip://${ip}:${port}，如ip://127.0.0.1:8000。

body为请求包体数据的json结构字符串，内部json字段要跟pb定义的字段完全一致。

begin开始事务：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/BeginTx -t
arget ip://127.0.0.1:8000 -body '{"key":"name", "value":"showerli"}'

[rsp json body]:{"msg":" ********begin******** "}
```

set值，成功添加：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Set -targe
t ip://127.0.0.1:8000 -body '{"key":"name", "value":"showerli"}'

[rsp json body]:{"msg":"set successfully"}
```

commit提交事务，后面无法继续提交：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Commit -ta
rget ip://127.0.0.1:8000 -body '{"key":"company", "value":"tecent"}'

[rsp json body]:{"msg":" ********commit success******** "}
```

尝试get获取value：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Get -targe
t ip://127.0.0.1:8000 -body '{"key":"name"}'

[rsp json body]:{"msg":"showerli"}
```

支持前缀索引：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Prefix -ta
rget ip://127.0.0.1:8000 -body '{"key":"na"}'

[rsp json body]:{"msg":"showerli , "}
```

支持回滚操作：

```
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/BeginTx -target ip://127.0.0.1:8000 -body '{"key":"name2
", "value":"liyuxiao"}'

[rsp json body]:{"msg":" ********begin******** "}
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Set -target ip://127.0.0.1:8000 -body '{"key":"name2", "
value":"liyuxiao"}'

[rsp json body]:{"msg":"set successfully"}
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Commit -target ip://127.0.0.1:8000 -body '{"key":"name2"
, "value":"liyuxiao"}'

[rsp json body]:{"msg":" ********commit success******** "}
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Rollback -target ip://127.0.0.1:8000 -body '{"key":"name
2", "value":"liyuxiao"}'

[rsp json body]:{"msg":" ********rollback success******** "}
[showerli@showerli1626920686854-0 ~]$ trpc-cli -func /trpc.kv.KVServer/Commit -target ip://127.0.0.1:8000 -body '{"key":"name2"
, "value":"liyuxiao"}'

[rsp json body]:{"msg":"no change found"}
```