hydra

后端全栈式服务框架，提供接口服务器、web服务器、websocket服务器，RPC服务器、统一调度服务器、消息消费服务器。并具有如下特点：

- 统一开发模式
  规范代码编写，采用统一方式编写服务
- 统一安装、启停、更新
  采用相同的方式进行系统初始化，服务启动、停止、热更新等
- 统一配置管理
  统一采用zookeeper 或 fs 保存配置,本地零配置。并采一方式进行配置安装
- 统一基础框架
  对缓存、数据库、消息队列、远程调用(GRPC)、日志归集、加解密等进行统一封装
- 统一日志框架
  集成本地日志、远程日志服务，对日志进行统一归集与编排。
- 统一服务治理
  集成服务注册、服务发现、负载均衡、流量控制、服务降级、熔断等，并对服务状态进行监控
- 统一认证服务
  可集成SAS远程认证服务集成，实现请求的访问权限控制、加解密等
- 统一监控报警
  集成系统监控、业务监控。对系统的主要指标(带宽、CPU、内存、硬盘、IO)和业务主要指标(QPS、处理时长、异常响应结果)进行统一采集并存入influxdb中，可使用grafana进行实时查看，与报警系统(convoy)集成可实现消息队列、数据库、服务状态等监控，和短信、微信等实时报警
- 多种集群模式
  支持集群分组及对等，主备，分片等集群模式
- golang 原生代码实现
- 20+线上项目实践经验

示例

1. api示例

新建文件夹 `hello` ,添加 `main.go` 文件,输入以下代码:

```
package main

import (
        "github.com/micro-plat/hydra/component"
        "github.com/micro-plat/hydra/hydra"
)

func main() {
        app := hydra.NewApp(
                hydra.WithPlatName("myplat"), //平台名称
                hydra.WithSystemName("demo"), //系统名称
                hydra.WithClusterName("test"), //集群名称
                hydra.WithServerTypes("api"), //服务器类型为http api
                hydra.WithRegistry("fs://../")) //使用本地文件系统作为注册中心

        app.API("/api",hello)
        app.Start()
}

func api(ctx *hydra.Context) (r interface{}) {
        return "hello world "
}
```

1. 编译生成服务

```
go install hello
```

2. 安装远程配置及本地服务

```
./hello install
```

3. 运行服务

```
./hello run
```

4. 测试服务

```
curl http://localhost:8090/hello

{"data":"hello world"}
```

2. RPC示例

   修改代码的服务器类型和注册方式即可

```go
package main

import (
        "github.com/micro-plat/hydra/component"
        "github.com/micro-plat/hydra/hydra"
)

func main() {
        app := hydra.NewApp(
                hydra.WithPlatName("myplat"), //平台名称
                hydra.WithSystemName("demo"), //系统名称
                hydra.WithClusterName("test"), //集群名称
                hydra.WithServerTypes("rpc"), //修改为RPC服务器类型
                hydra.WithRegistry("fs://../")) //使用本地文件系统作为注册中心

        app.RPC("/rpc",hello) //注册为rpc服务
        app.Start()
}

func rpc(ctx *hydra.Context) (r interface{}) {
        return "hello world "
}
```

测试上述代码，可在api服务中直接调用RPC服务

```go
func api(ctx *hydra.Context) (r interface{}) {
        _, result, _, err := ctx.RPC.Request("/rpc", nil, nil, true)
        if err != nil {
                return err
        }
        return result
}
```

也可以启动多个服务器

```go
package main

import (
        "github.com/micro-plat/hydra/hydra"
)

func main() {
        app := hydra.NewApp(
                hydra.WithPlatName("myplat"),      //平台名称
                hydra.WithSystemName("demo"),      //系统名称
                hydra.WithClusterName("test"),     //集群名称
                hydra.WithServerTypes("api-rpc"), //多个服务器类型，使用"-"分隔
                hydra.WithRegistry("fs://../"))    //使用本地文件系统作为注册中心

        app.API("/api", api)
        app.RPC("/rpc", rpc) //注册为rpc服务
        app.Start()
}

func rpc(ctx *hydra.Context) (r interface{}) {
        return "hello world "
}
func api(ctx *hydra.Context) (r interface{}) {
        _, result, _, err := ctx.RPC.Request("/rpc", nil, nil, true)
        if err != nil {
                return err
        }
        return result
}
```

编译，安装，并启动,测试服务，可看到如下日志：

```
[2020/01/06 16:51:56.924002][i][1bfd99afb]初始化 /myplat/demo/api-rpc/test
[2020/01/06 16:51:56.925285][i][8b4b78124]开始启动[API]服务...
[2020/01/06 16:51:56.925598][d][8b4b78124][未启用 jwt设置]
[2020/01/06 16:51:56.925606][d][8b4b78124][未启用 ajax请求限制设置]
[2020/01/06 16:51:56.925583][i][8b4b78124][启用 静态文件]
[2020/01/06 16:51:56.925600][d][8b4b78124][未启用 固定密钥认证]
[2020/01/06 16:51:56.925590][d][8b4b78124][未启用 header设置]
[2020/01/06 16:51:56.925609][d][8b4b78124][未启用 metric设置]
[2020/01/06 16:51:56.925593][d][8b4b78124][未启用 响应格式设置]
[2020/01/06 16:51:56.925604][d][8b4b78124][未启用 远程服务认证]
[2020/01/06 16:51:56.925611][d][8b4b78124][未启用 host设置]
[2020/01/06 16:51:56.925596][d][8b4b78124][未启用 熔断设置]
[2020/01/06 16:51:57.426422][i][8b4b78124]服务启动成功(API,http://192.168.4.121:8090,1)
[2020/01/06 16:51:57.426655][i][ec80eadba]开始启动[RPC]服务...
[2020/01/06 16:51:57.427484][d][ec80eadba][未启用 远程服务认证]
[2020/01/06 16:51:57.427500][d][ec80eadba][未启用 metric设置]
[2020/01/06 16:51:57.427452][d][ec80eadba][未启用 jwt设置]
[2020/01/06 16:51:57.427492][d][ec80eadba][未启用 header设置]
[2020/01/06 16:51:57.427472][d][ec80eadba][未启用 固定密钥认证]
[2020/01/06 16:51:57.427505][d][ec80eadba][未启用 host设置]
[2020/01/06 16:51:57.928270][i][ec80eadba]服务启动成功(RPC,tcp://192.168.4.121:8081,1)
[2020/01/06 16:52:18.287947][i][979a5d210]api.request GET /api from 192.168.4.121
[2020/01/06 16:52:19.290044][i][979a5d210]rpc.request: myplat.demo.test GET /rpc from
[2020/01/06 16:52:19.290626][i][979a5d210]rpc.response: myplat.demo.test GET /rpc 200  6
[2020/01/06 16:52:19.291206][i][979a5d210]api.response GET /api 200  1.003274125s
```

其它示例请查看以下文档

hydra 微服务开发入门

hydra 微服务开发规范

构建 API 服务一

构建 API 服务二

构建消息消费服务(MQC)

构建任务调度服务(CRON)

构建远程调用服务(RPC)

构建静态文件服务

构建 websocket 服务(WS)

安全认证(jwt,数字摘要，远程认证)

性能指标收集(metric)

日志配置与管理