**[学习微服务第9天] Service-Center使用入门**

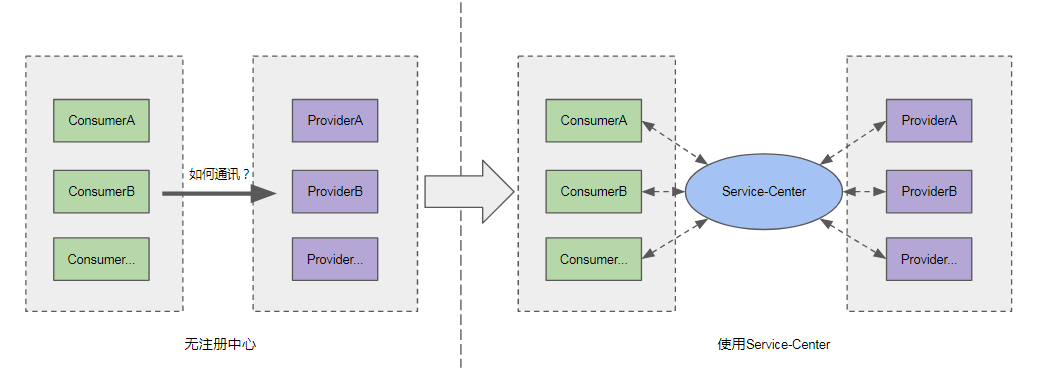
上一篇我们介绍了ServiceComb内置的负载均衡组件handler-loadbalance，本篇我们将介绍Service-Center使用入门。

一. Service-Center 是什么

ServiceCenter是一个服务注册中心。服务提供者可以将自身的实例信息注册到 ServiceCenter，以供服务消费者发现并使用它。

二. 为什么使用Service-Center

在微服务架构中，一个应用由一组职责单一化的服务组成，各个服务被动态的部署到不同的节点。面对这样一组服务，应该如何去管理服务之间的依赖关系呢？



服务注册中心的出现正是为了解决这样的问题，它提供的注册机制，允许服务提供者将自己的信息登记到中心；提供的发现机制，供服务消费者从中心查找服务提供者信息。

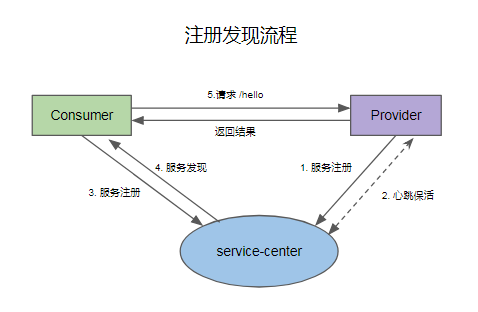
服务注册中心优点：

1.解耦服务提供者与服务消费者，服务消费者不需要硬编码服务提供者地址。

2.服务动态发现及可伸缩能力，服务提供者实例的动态增减能通过注册中心动态推送到服务消费者端。

3.通过注册中心可以动态的监控服务运行质量及服务依赖，为服务提供服务治理能力。

三. 注册发现流程



如上图，Service-Center中服务发现流程大致有以下几个步骤：

* 服务提供者向Service-Center注册服务信息
* 服务提供者发送心跳，维持在Service-Center中的“UP”状态
* 服务消费者向Service-Center注册服务信息
* 服务消费者从Service-Center发现服务提供者信息
* 服务消费者向服务提供者发送请求，并获取通讯结果

•Service-Center注册发现接口基于RESTful标准实现，不受开发语言限制，实现对应接口可以参考官方API文档：

https://rawcdn.githack.com/ServiceComb/service-center/master/docs/api-docs.html

•Service-Center提供了简单注册与发现的Client，其中封装了API实现，可直接使用，具体可查看：

https://github.com/apache/servicecomb-service-center/tree/master/pkg/client/sc

四. 使用Service-Center

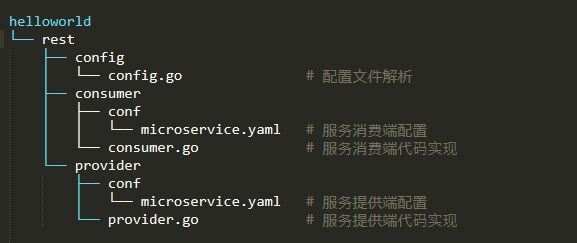
为了更好的理解流程，下面我们将通过“helloword”的示例，实现基于Service-Center的注册发现，并完成Consumer与Provider之间的通讯。

以下仅展示了主要流程代码，完整示例请参考：

https://github.com/ChinX/service-center-demo/tree/simple-demo

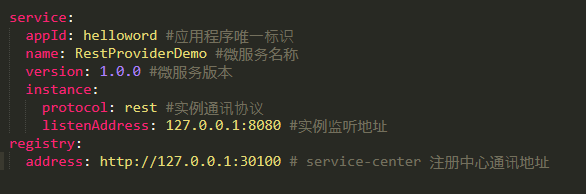
1.目录参考

创建名为“helloworld”的项目，以下为参考目录结构：



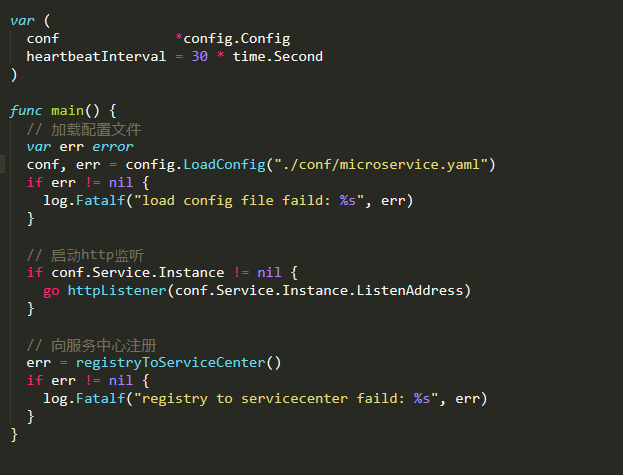
2. 服务提供端实现

新建服务提供端配置文件： helloworld/rest/provider/conf/microservice.yaml

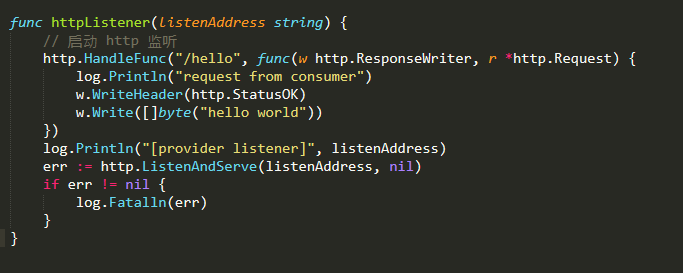


新建项目入口文件： helloworld/rest/provider/provider.go

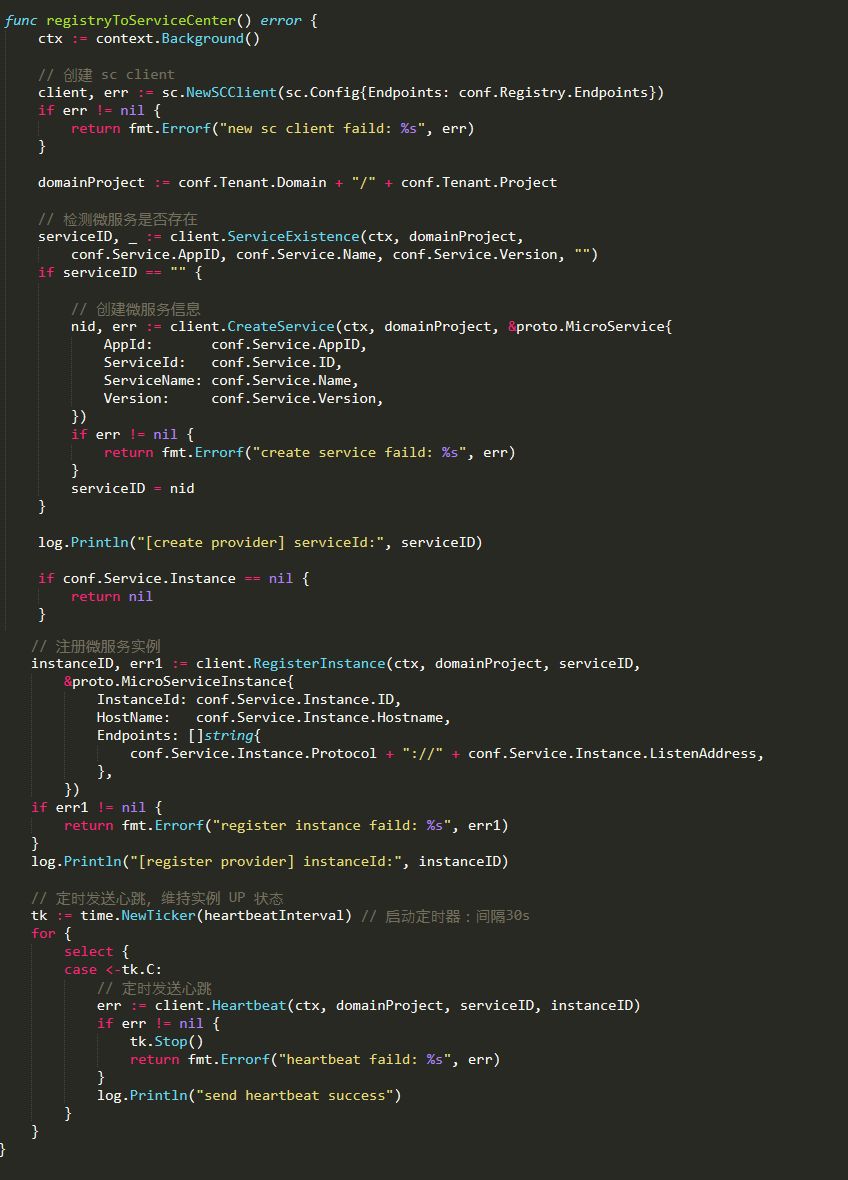
1.在main函数中完成启动流程：



2.启动服务提供端http监听

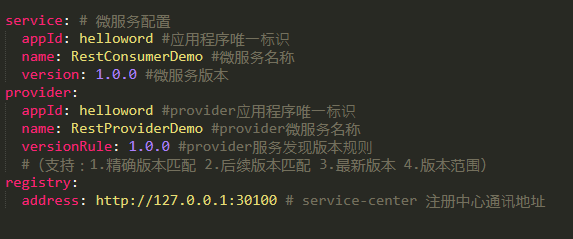


3.向Service-Center注册自身服务，其中包含微服务创建、实例注册、心跳保活三个部分，具体代码如下：



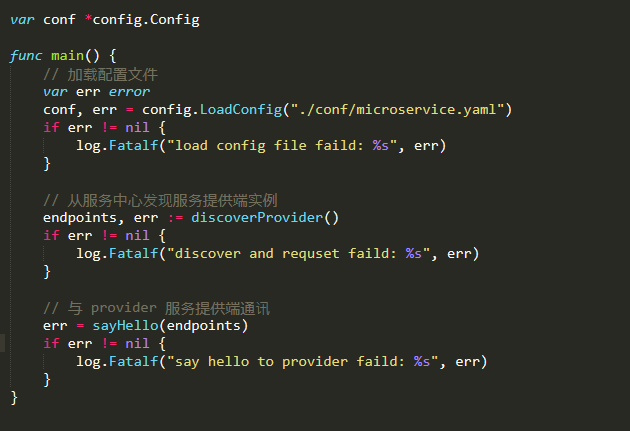
3. 服务消费端实现

配置文件： helloworld/rest/provider/conf/microservice.yaml

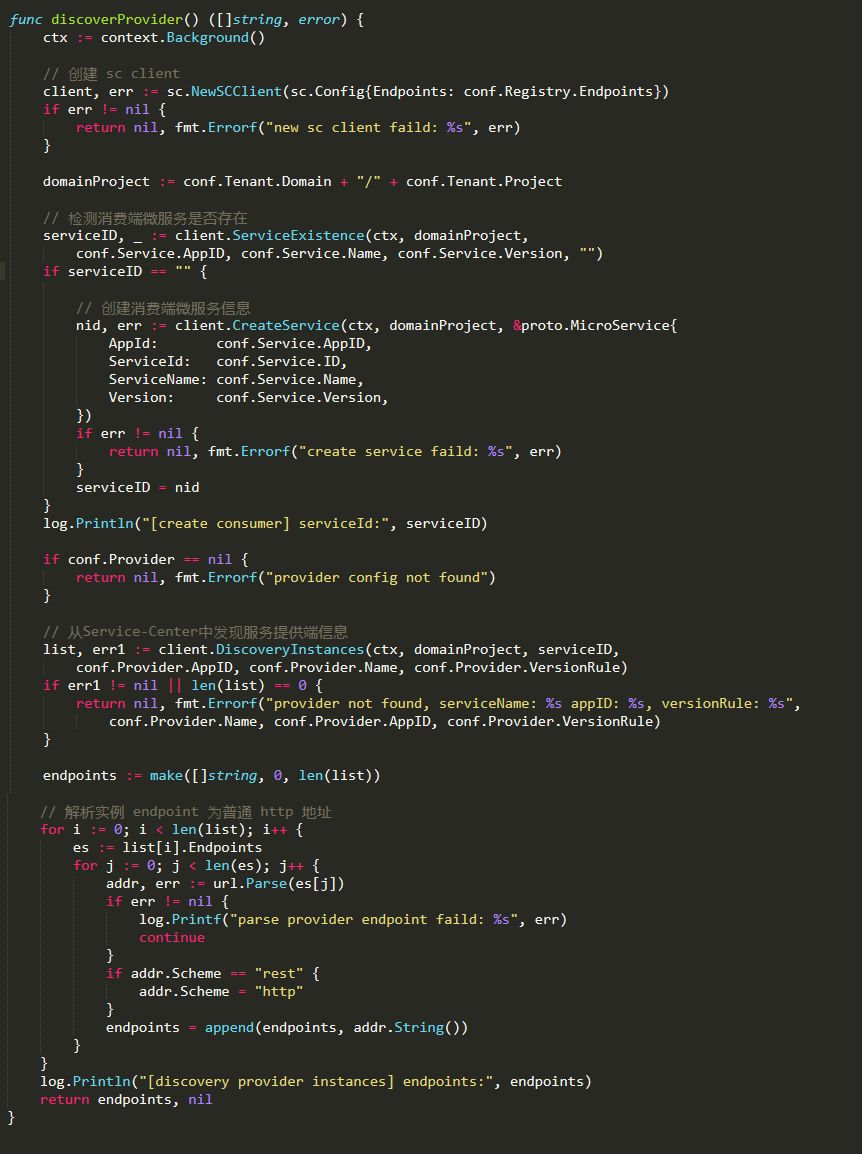


入口文件： helloworld/rest/consumer/consumer.go

1.在main函数中完成启动流程：



2.从Service-Center服务发现服务提供端,涉及创建服务、服务发现接口：



3.请求接口“/hello”完成与服务提供端的通讯



4. 构建编译

以下基于 go 1.11+ 进行构建，请检测自身 go 环境。  
进入 service-center-demo 目录:

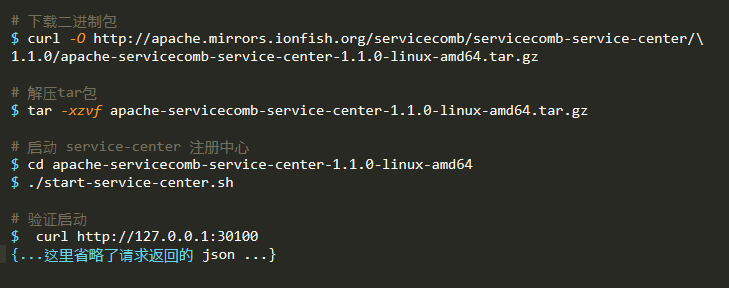


五. 功能验证

1. 安装启动 Service-Center

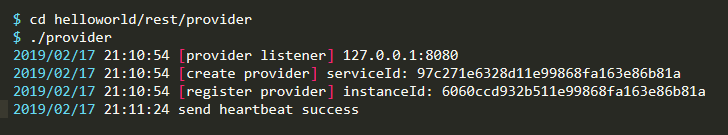
从Service-Center官网获取二进制包：

http://servicecomb.apache.org/release/ ，解压并运行二进制



执行如上命令并得到对应的反馈信息，则说明安装部署成功。

2. 启动 provider



输出信息分析：

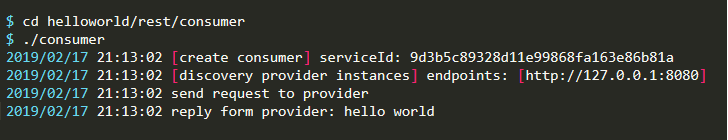
服务提供端启动并监听了本地8080端口。

成功创建微服务服务，并打印了返回的serviceId。

成功注册自身实例，并打印了返回的instanceId。

服务提供端在启动30秒后，成功发送了一次心跳。

3. 启动 consumer



输出信息分析：

服务消费端成功创建微服务服务，并打印了返回的serviceId。

服务消费端成功发现provider实例，并打印了返回的provider endpoints。

服务提供端成功向服务提供端发送请求，并获得返回的信息“hello world”。

此时查看provider端控制台，若出现“2019/02/17 21:13：02 request from consumer”类似字样的打印，则说明provider端接收到consumer端的请求，共同印证了通讯的成功。