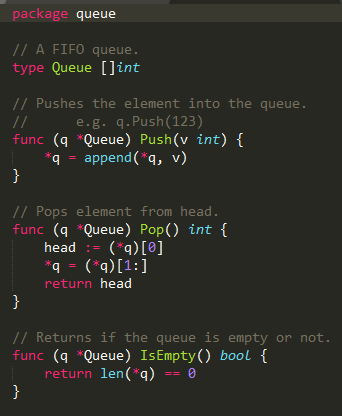


第一种--别名：



安装goimports工具

go get -v github.com/gpmgo/gopm

进入/c/gocode/src/github.com/gpmgo/gopm

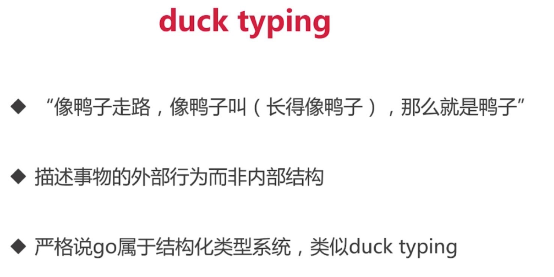
gopm get -g -v -u golang.org/x/tools/cmd/goimports

go install golang.org/x/tools/cmd/goimports

第二种--组合：



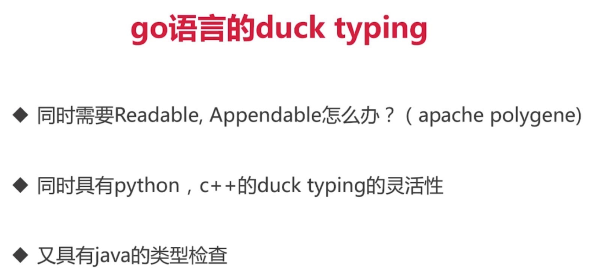






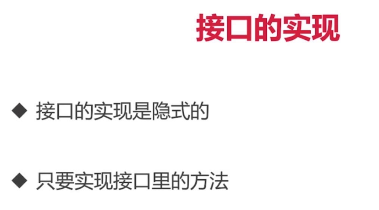


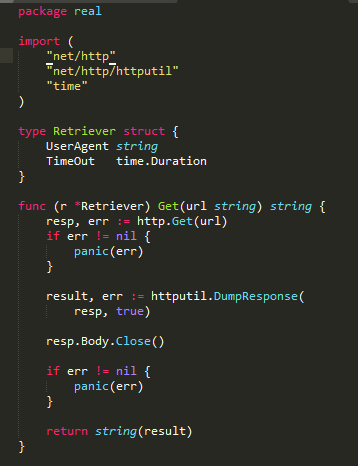






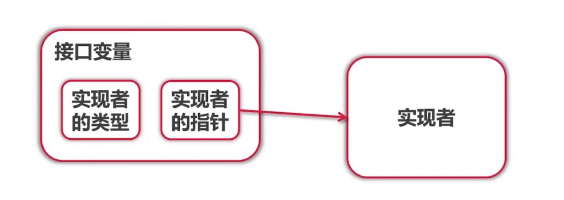


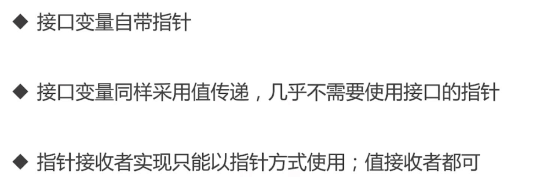


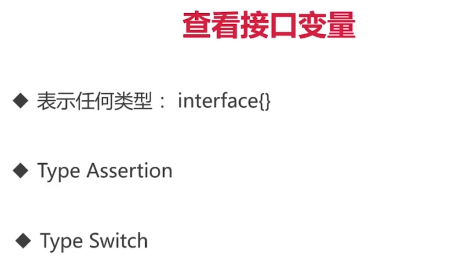


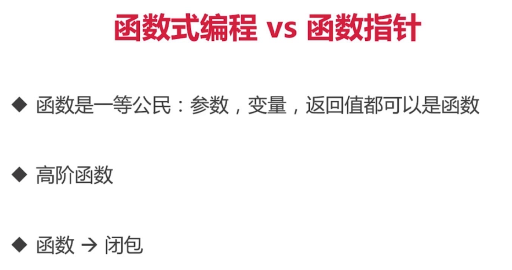


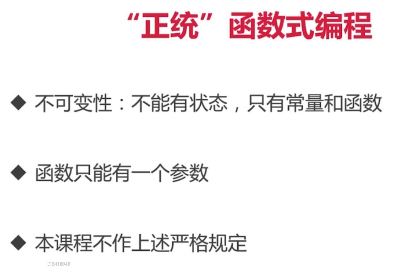


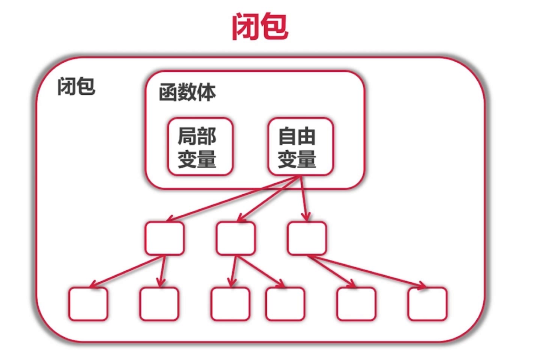


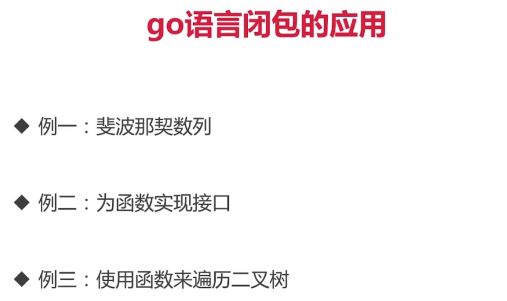


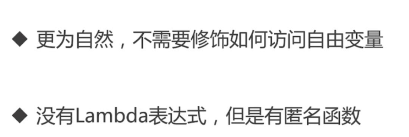


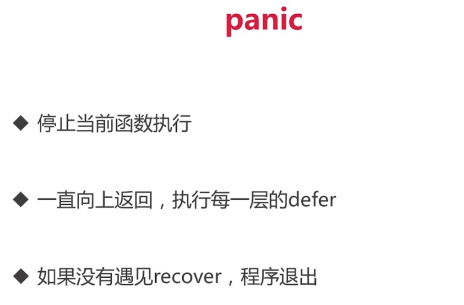




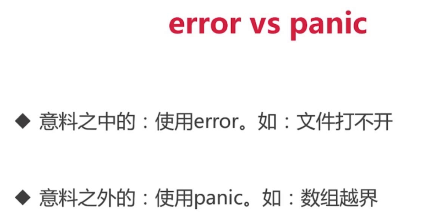


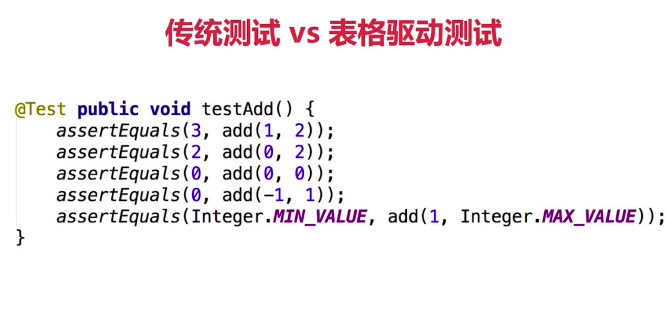


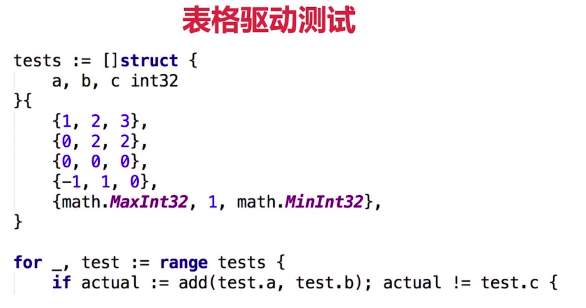


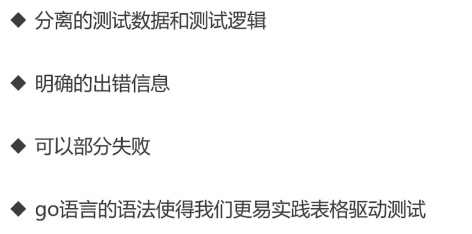












测试覆盖率

go test -coverprofile=c.out

go tool cover

go tool cover -html=c.out

基准测试：

go test –bench=.

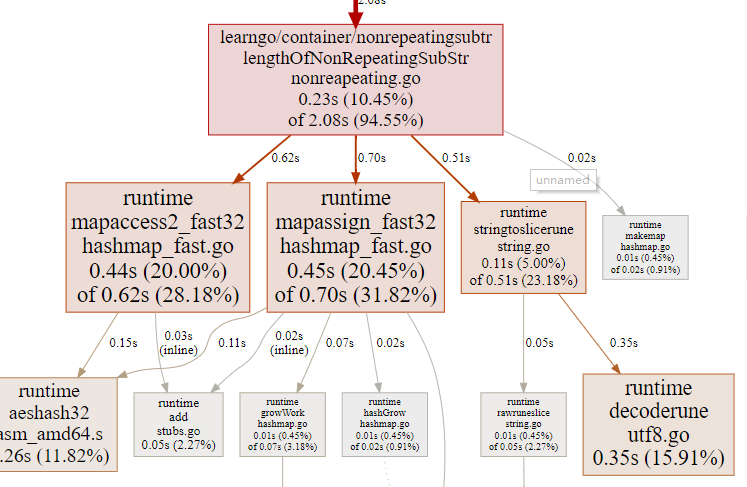
性能分析（需要安装https://graphviz.gitlab.io/\_pages/Download/Download\_windows.html）

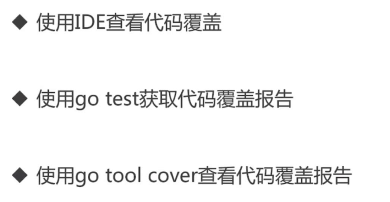
go test -bench . -cpuprofile cpu.out

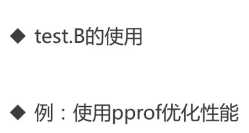
go tool pprof cpu.out

web

quit









go doc

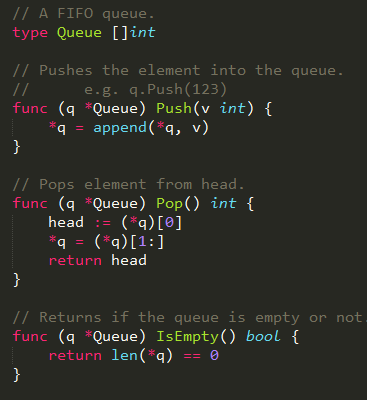
go doc Queue

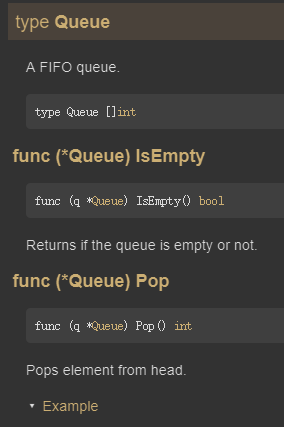
go help doc

go doc fmt.Println

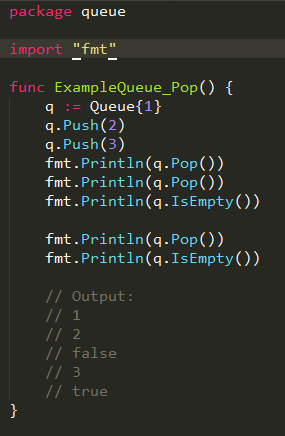
godoc -http :6060

（在http://localhost:6060/pkg/project/u2pppw/queue/查看文档，本地写的都可以看到）





写实例代码：

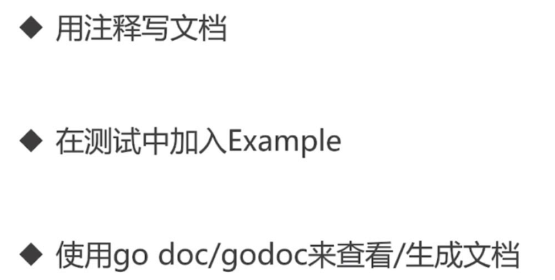


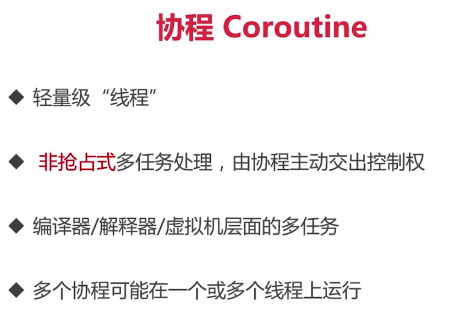
可以先写 //Output:

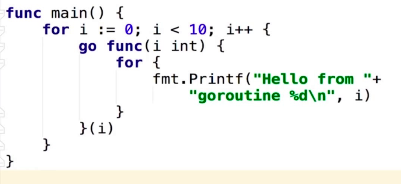
//2

得出结果后再补全,文档会出现写好的测试用例

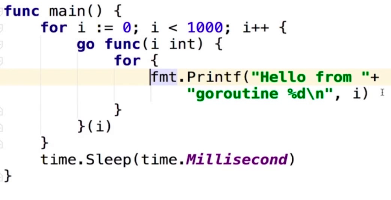




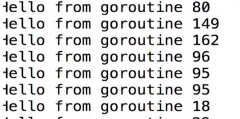




上面的程序，还没来得及打印就退出了，原因是main()所在的主goroutine已经执行完毕了



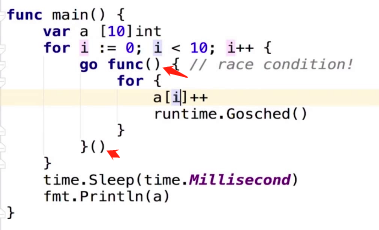
结果：



上面的程序中，fmt.Printf()是I/O操作，会有等待和切换操作，所以一个goroutine在打印到一半就会跳出来，然后轮到其他的goroutine打印

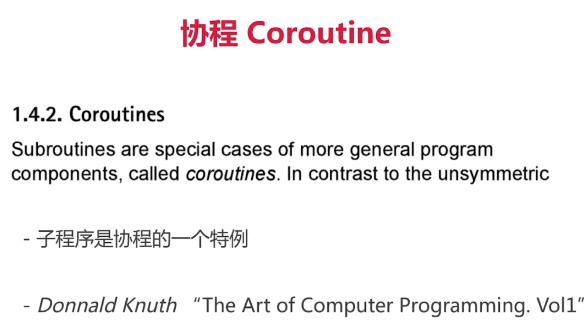


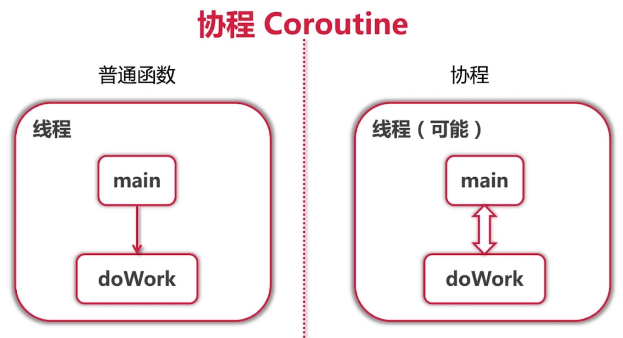
上面的程序运行后会死机，cpu占用率非常高，a[i]++被一个goroutine抢占，没有主动交出控制权，main函数也是一个goroutine，它要把i传进去给数组a因为没人交出控制权，也卡在那里，永远执行不到time.Sleep()这一步，使用runtime.Gosched()可以主动交出控制权

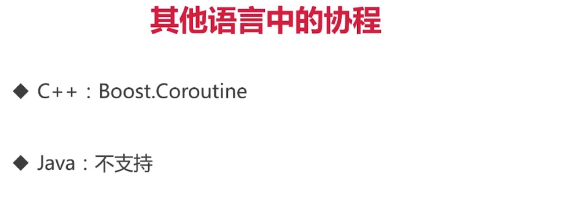


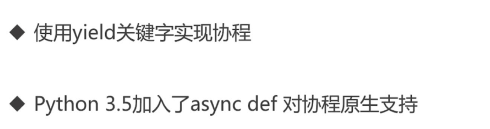
如果不显式传i，那么闭包的i是引用自由变量i的，所以用的i都是10，最后产生越界的错误

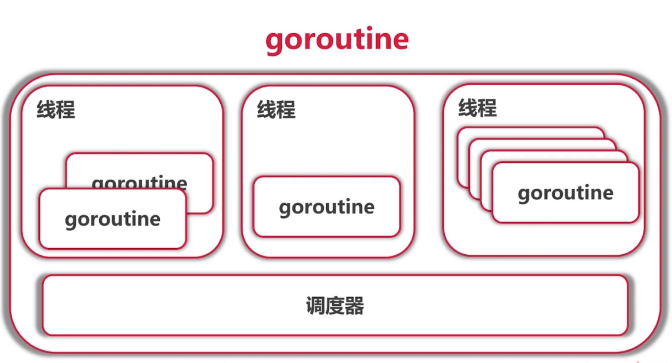
go run –race test.go可以检查冲突

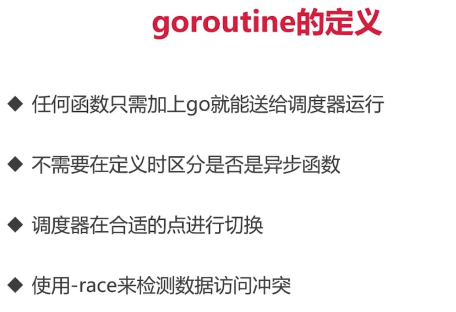




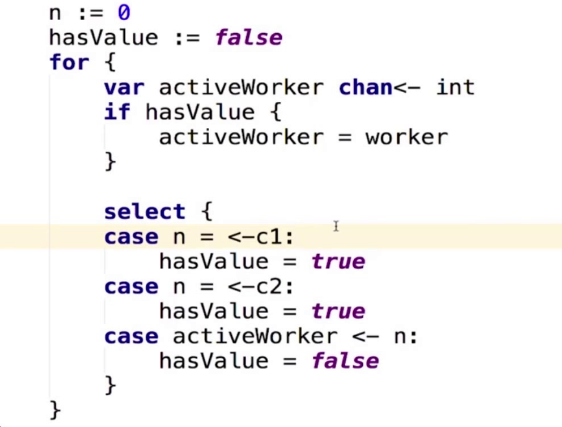


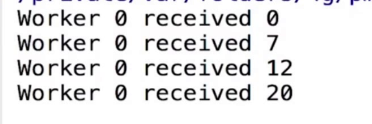












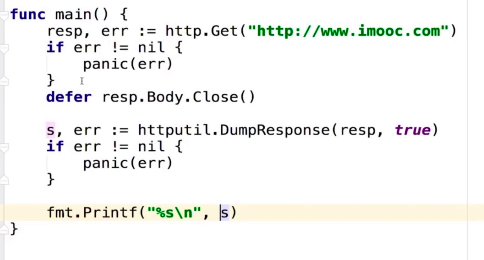
n 生成的速度太快，消耗的速度太慢会导致一部分n被冲掉了

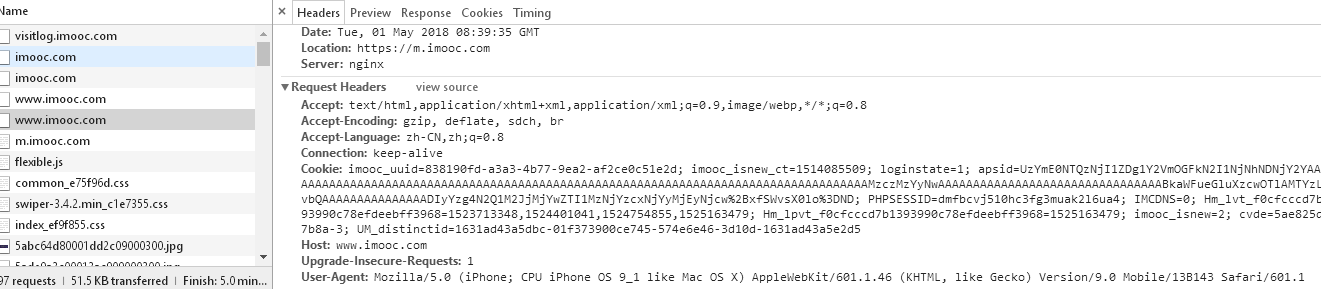






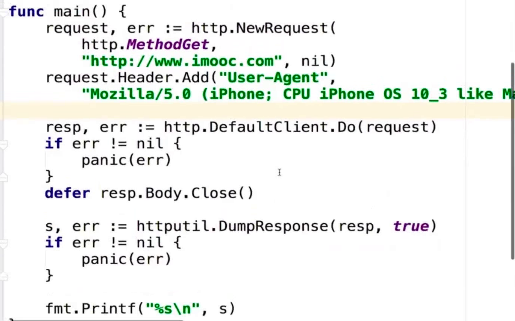


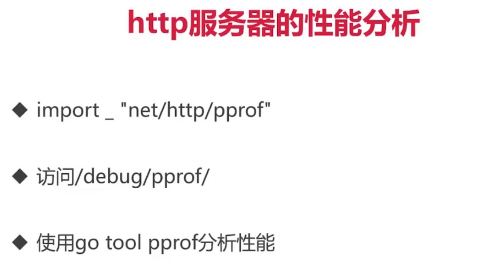




User-Agent:Mozilla/5.0 (iPhone; CPU iPhone OS 9\_1 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13B143 Safari/601.1

获取手机版的imooc，设置请求头





例如：http://localhost:8888/debug/pprof/

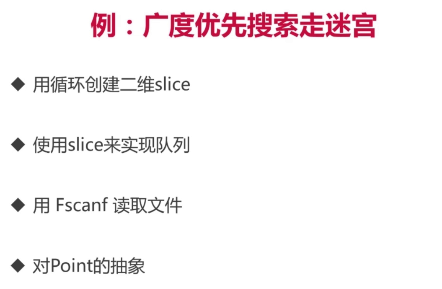
例如：go tool pprof http://localhost:8888/debug/pprof/profile 查看30秒内CPU使用情况

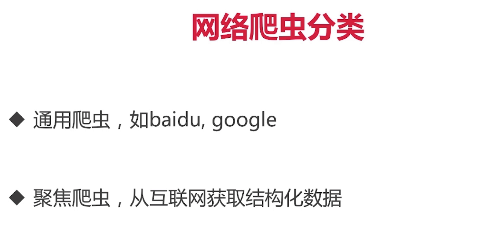
web 查看svg

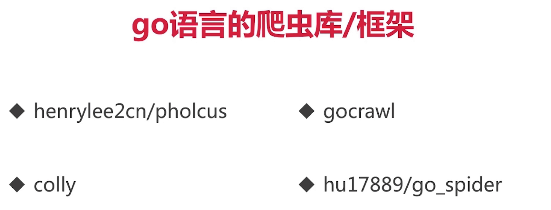
在pprof.go里面有明:

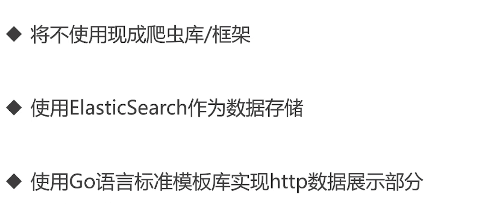


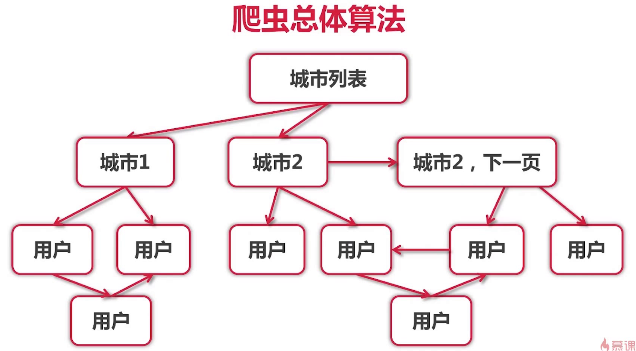
例如：go tool pprof http://localhost:8888/debug/pprof/heap内存使用情况







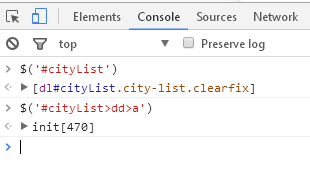






gopm get -g -v golang.org/x/text 用来转码

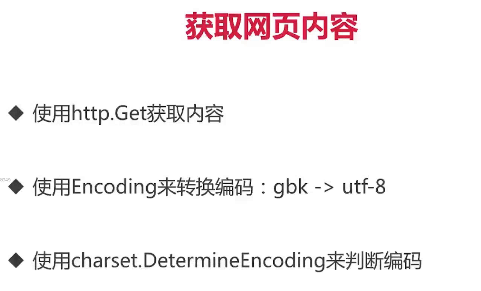


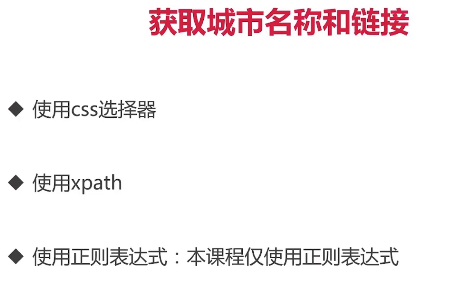


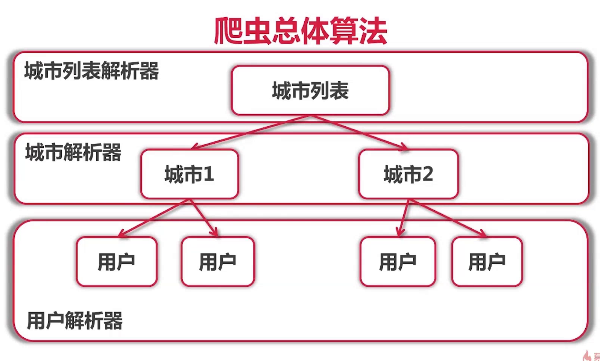


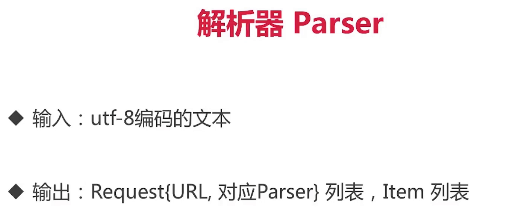
上面是css选择器

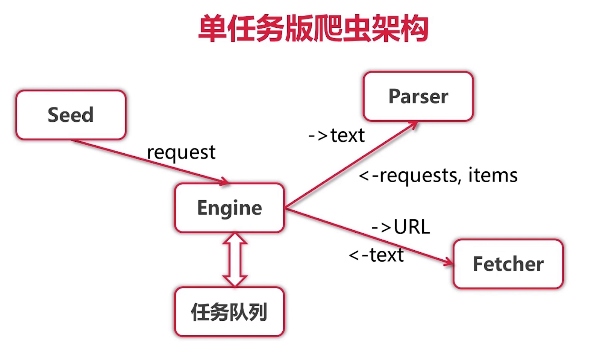




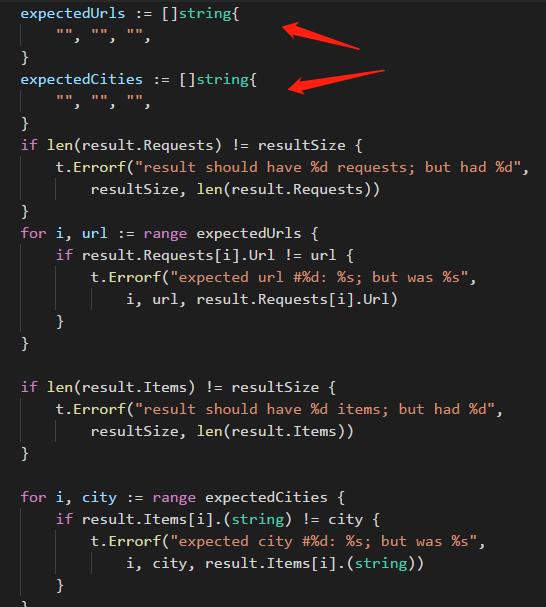


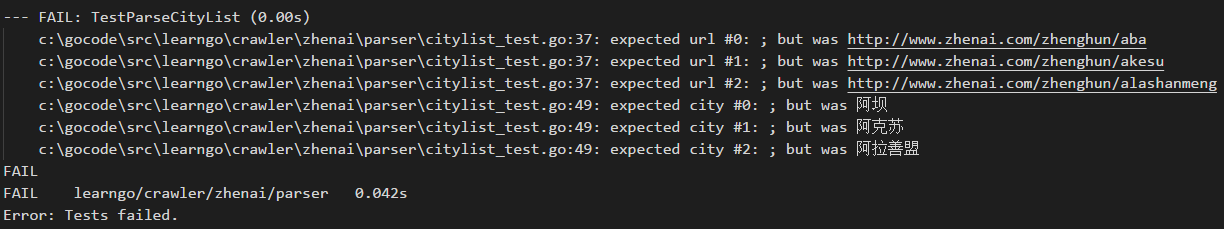


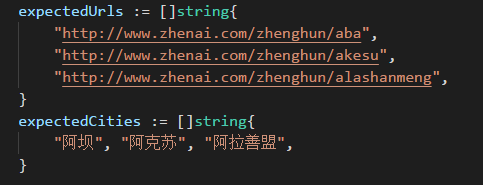


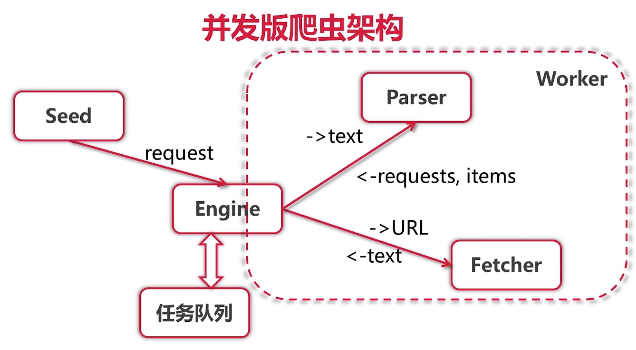


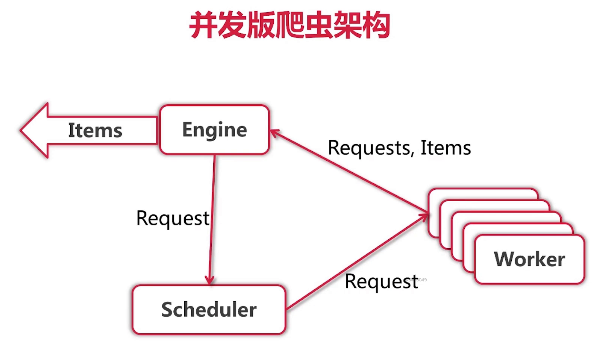
测试可以先这样,把它设为空，得到结果再填上去

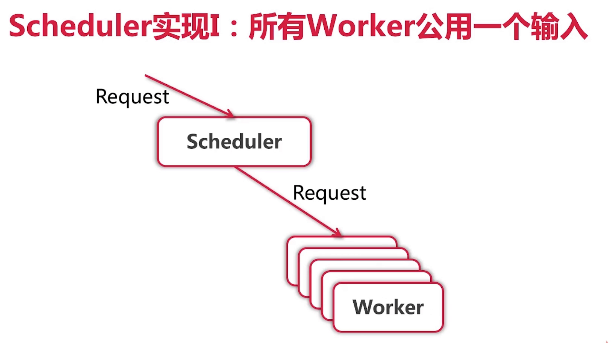


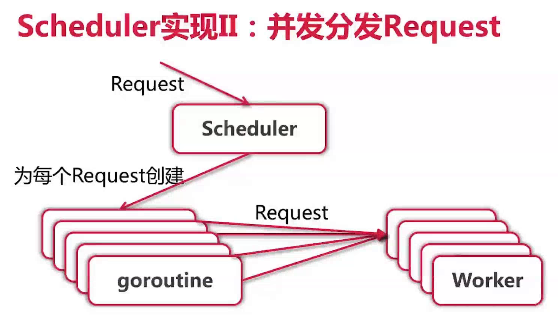


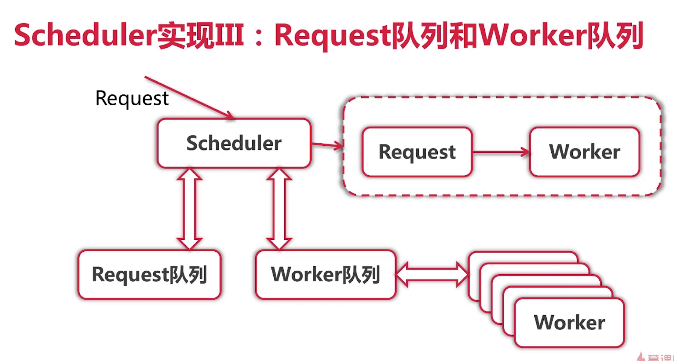


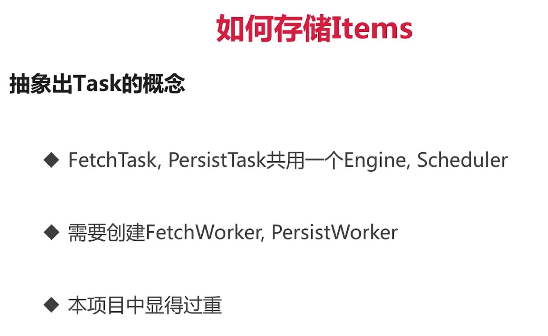


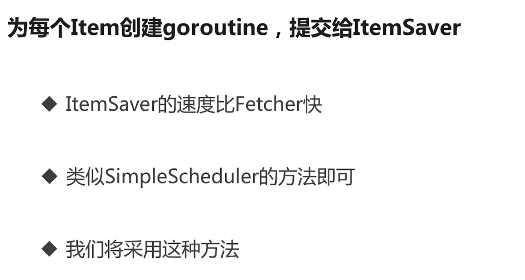


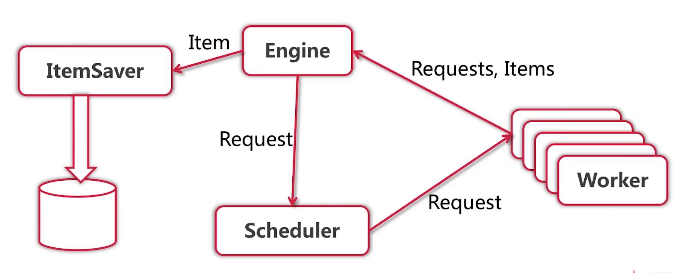






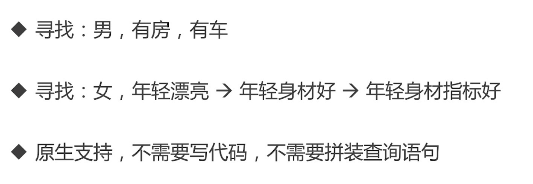


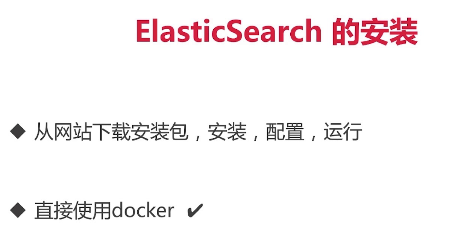


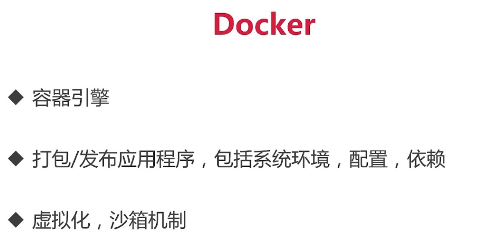


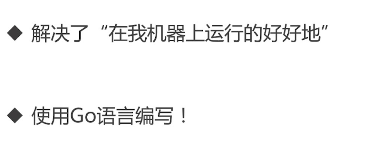


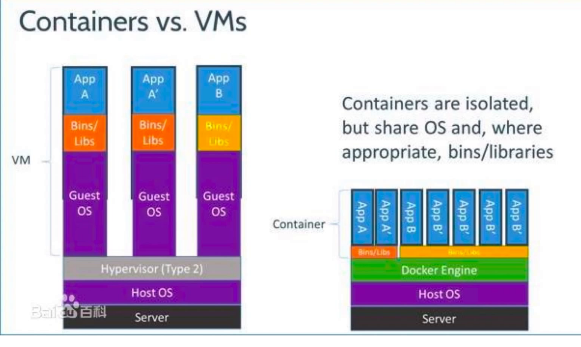










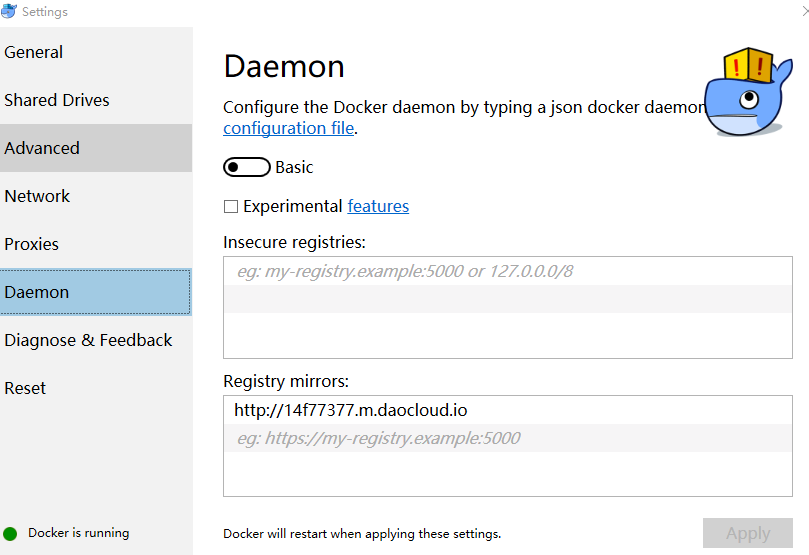


docker version

docker info

docker run hello-world

打开<http://www.daocloud.io/mirror>使用加速器



运行 docker info 命令发现



下面用power shell的命令行

docker run -it alpine sh 运行一个最小的Linux镜像

exit 退出镜像

docker images 查看拉取过的镜像

docker run –d –p 80:80 nginx 运行nginx镜像，-d是在后台运行, -p 80:80把容器的80端口映射到物理主机的80端口

docker ps 可以看到nginx在后台运行了

访问<http://localhost/:80> 即可看到nginx正在运行

docker kill 3a0881a5842f 即可关闭

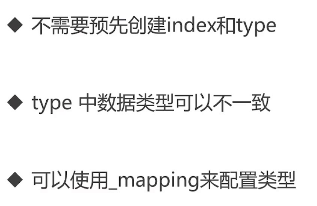
**正式开始elasticsearch的运行：**

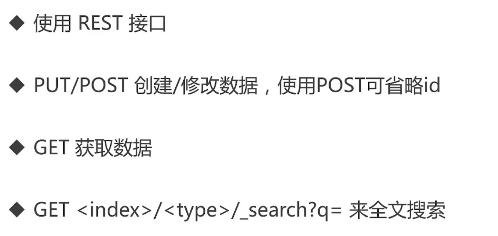
docker run -d-p 9200:9200 elasticsearch

docker ps

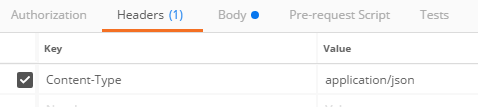
docker logs 55e1f10b0fc0 查看日志



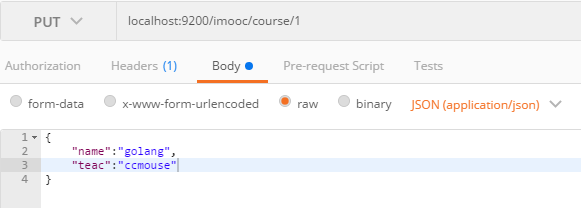




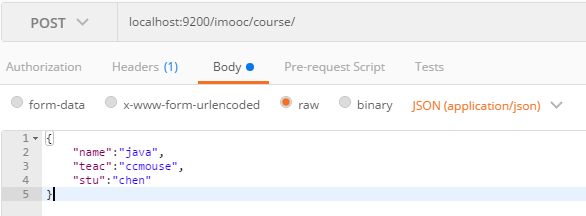
例子：



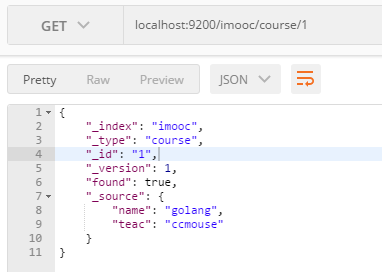
put或者post都可以用来创建新纪录和更新记录，格式和下面一致



用post可以不加id，另外字段可以不保持一致，得到结果的id不是有序的数字



查看结果：



GET localhost:9200/imooc/course/\_search 搜索全部记录

GET localhost:9200/imooc/course/\_search?q=golang 命中关键词golang的记录

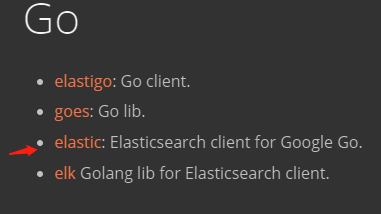
GET localhost:9200/imooc/course/\_mapping 查看字段类型

使用elasticsearch的客户端

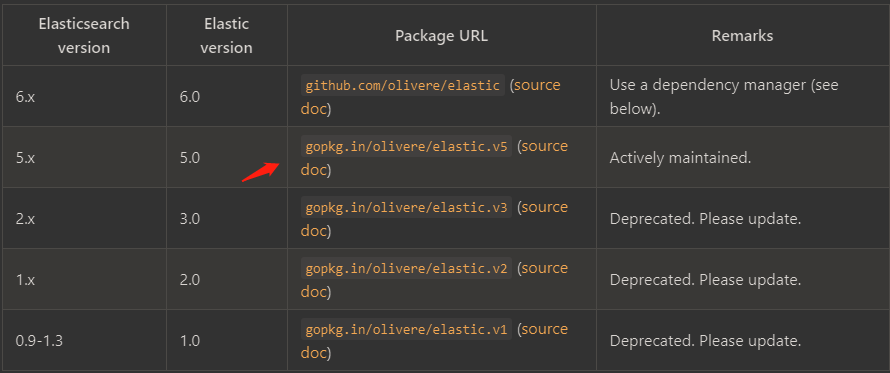
[www.elastic.co/guide/en/elasticsearch/client/community/current/index.html#go](http://www.elastic.co/guide/en/elasticsearch/client/community/current/index.html#go)

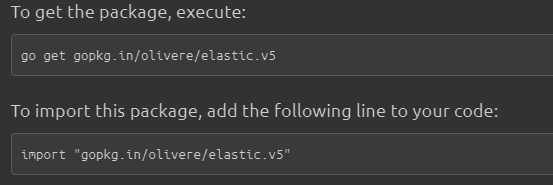
docker kill 55e1f10b0fc0 关闭后可删掉测试存的数据

docker run -d-p 9200:9200 elasticsearch 重新开始



选择合适的版本(访问localhost:9200即可看到版本信息)

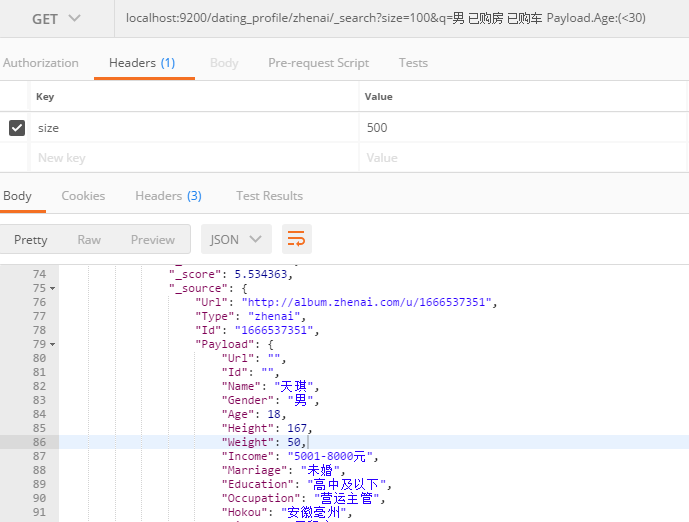


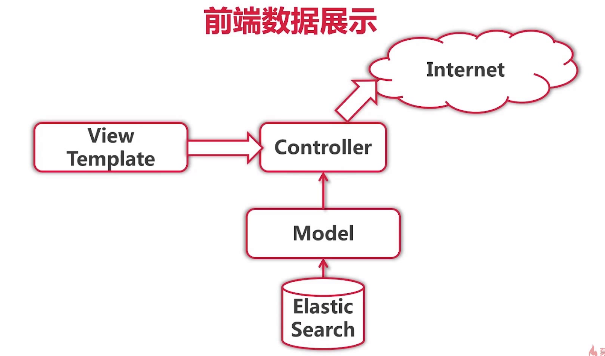


localhost:9200/dating\_profile/zhenai/\_search?

size=100&q=男 已购房 已购车 Age:(<30) Height([178 TO 185])

注意：输入是否对Payload.Age做了处理





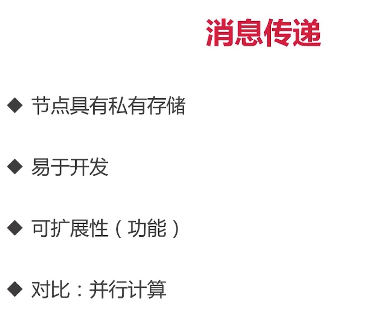


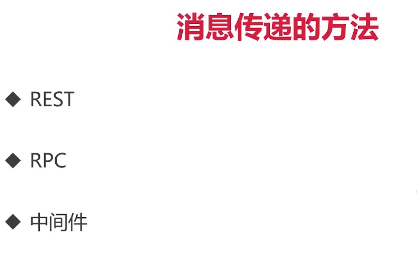


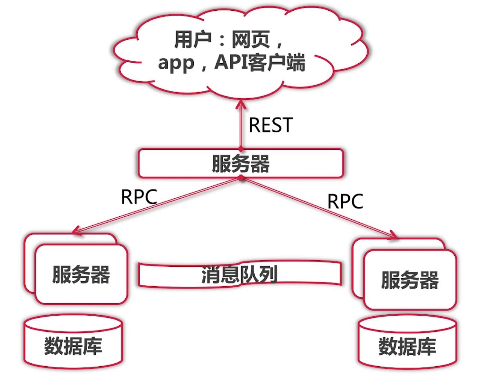




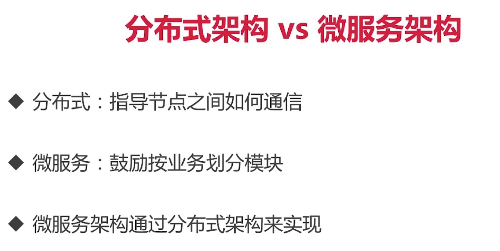


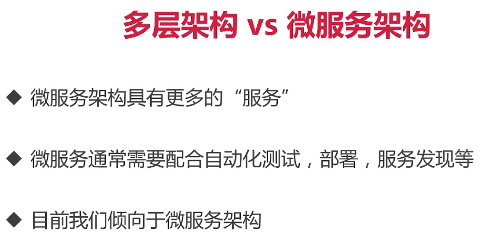


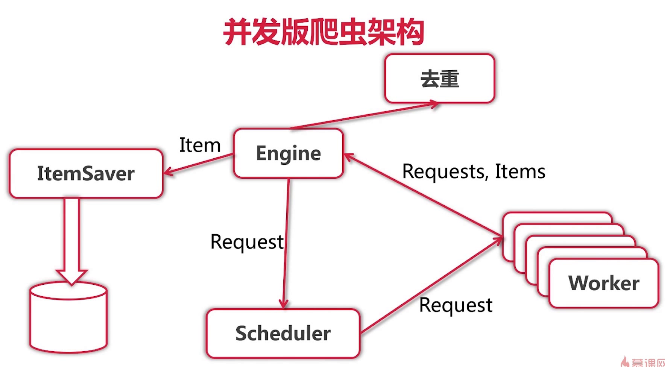




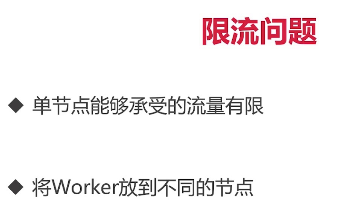


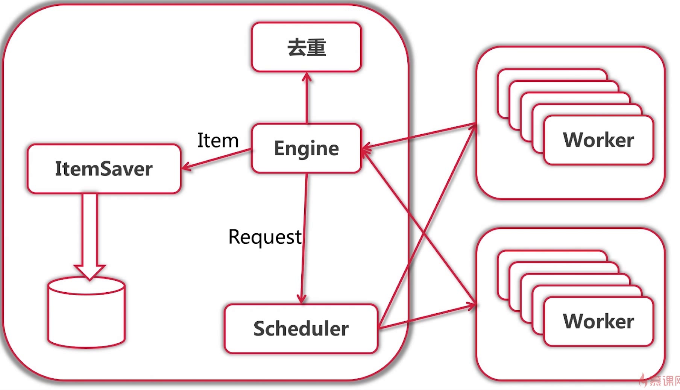




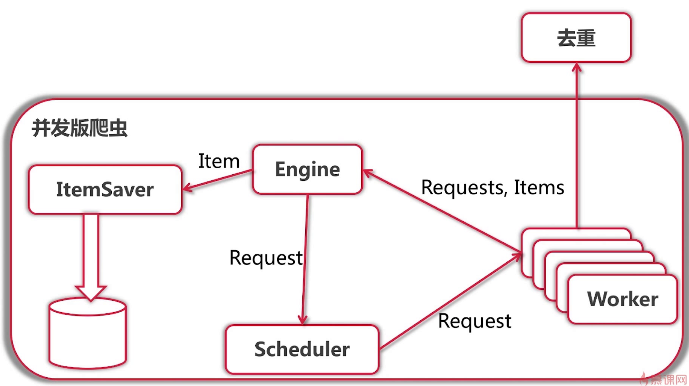


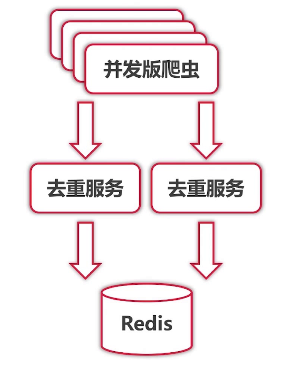


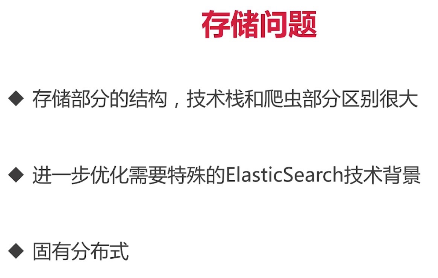


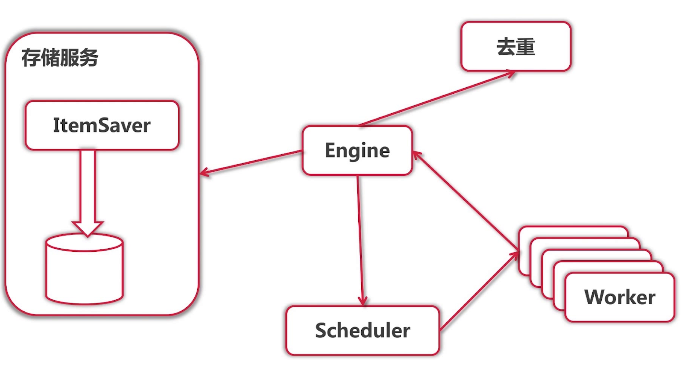


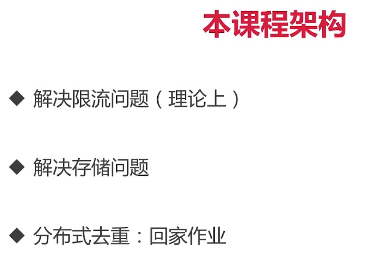


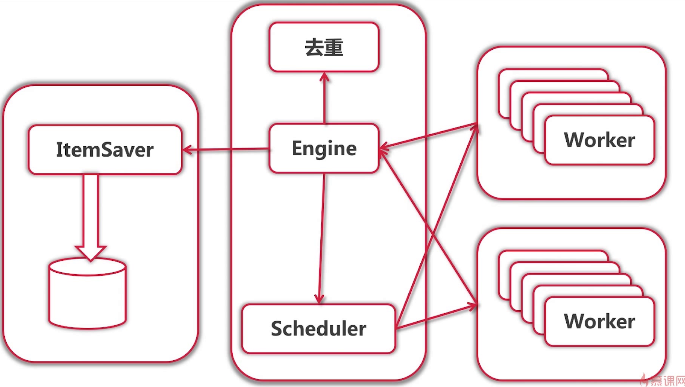


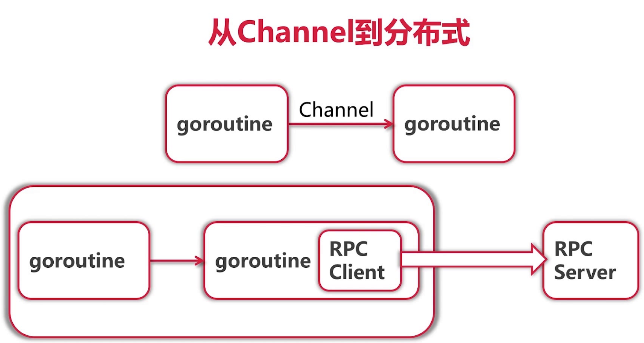












本课程采用上面这种分布式



