Go在Docker分布式环境中的应用

孙宏亮@DaoCloud

allen.sun@daocloud.io

个人介绍

- 孙宏亮
- DaoCloud技术合伙人，高级工程师
- 热爱golang&docker
- 《Docker源码分析》作者
- docker、swarm等项目committer

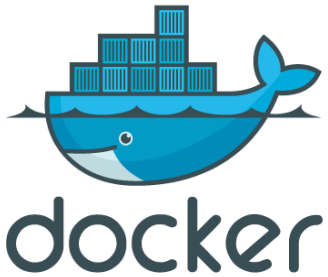
Agenda

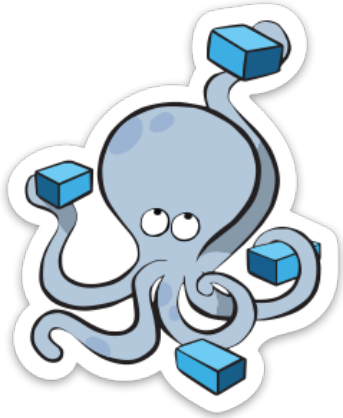
- Docker生态&Golang
- DaoCloud&Golang
- Docker运维&Golang
- 总结

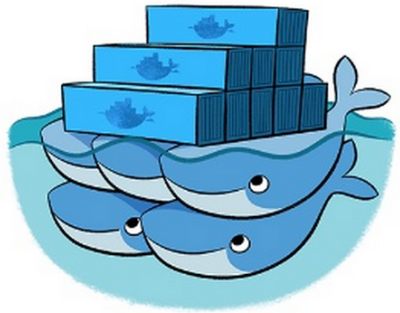
Docker生态

广义的Docker，多代表生态

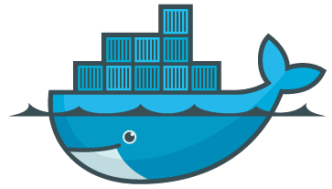
- Docker公司营造的整个生态
  - 容器与镜像：Docker
  - 容器编排与部署能力：Compose
  - 容器集群管理：Swarm
  - 容器底层的机器管理：Machine
- 容器市场生态
  - CoreOS与Docker，以及Rocket
  - Kubernetes与Docker：容器编排能力
  - Mesos与Docker：资源管理

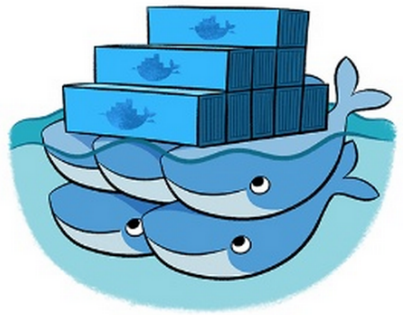






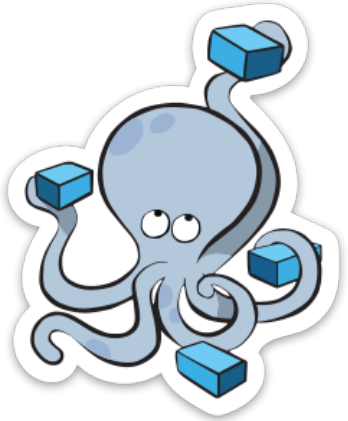
Golang&Docker生态





golang

python



C

DaoCloud&Golang

DaoShip

分布式持续集成流水线

持续集成
1.对接代码托管平台
2.自定义集成规则
3.执行镜像构建

DaoHub

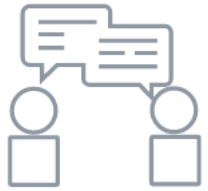
企业级容器镜像仓库

镜像仓库
1.用户认证
2.镜像托管
3.镜像高可用存储

DaoCloud

企业级容器云平台

PaaS平台
1.应用生命周期管理
2.弹性能力
3.混合云能力



DaoVoice

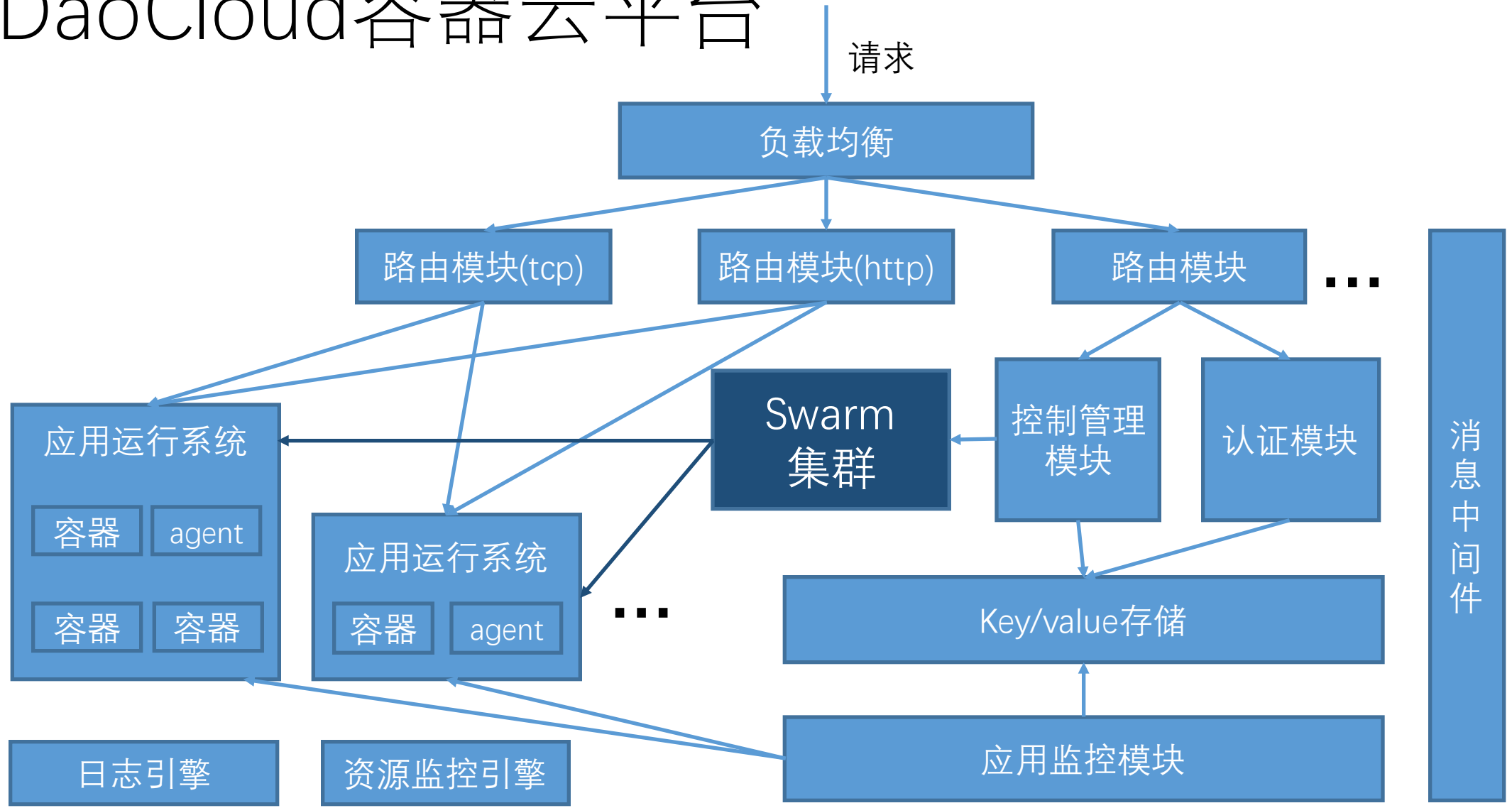
数据驱动精益运营

数据运营
1.SaaS服务
2.用户数据采集
3.数据分析
4.驱动产品运营

DaoCloud容器云平台

- 负载均衡
- 动态路由
- Swarm容器调度
- 容器监控
- 应用监控
- 消息中间件

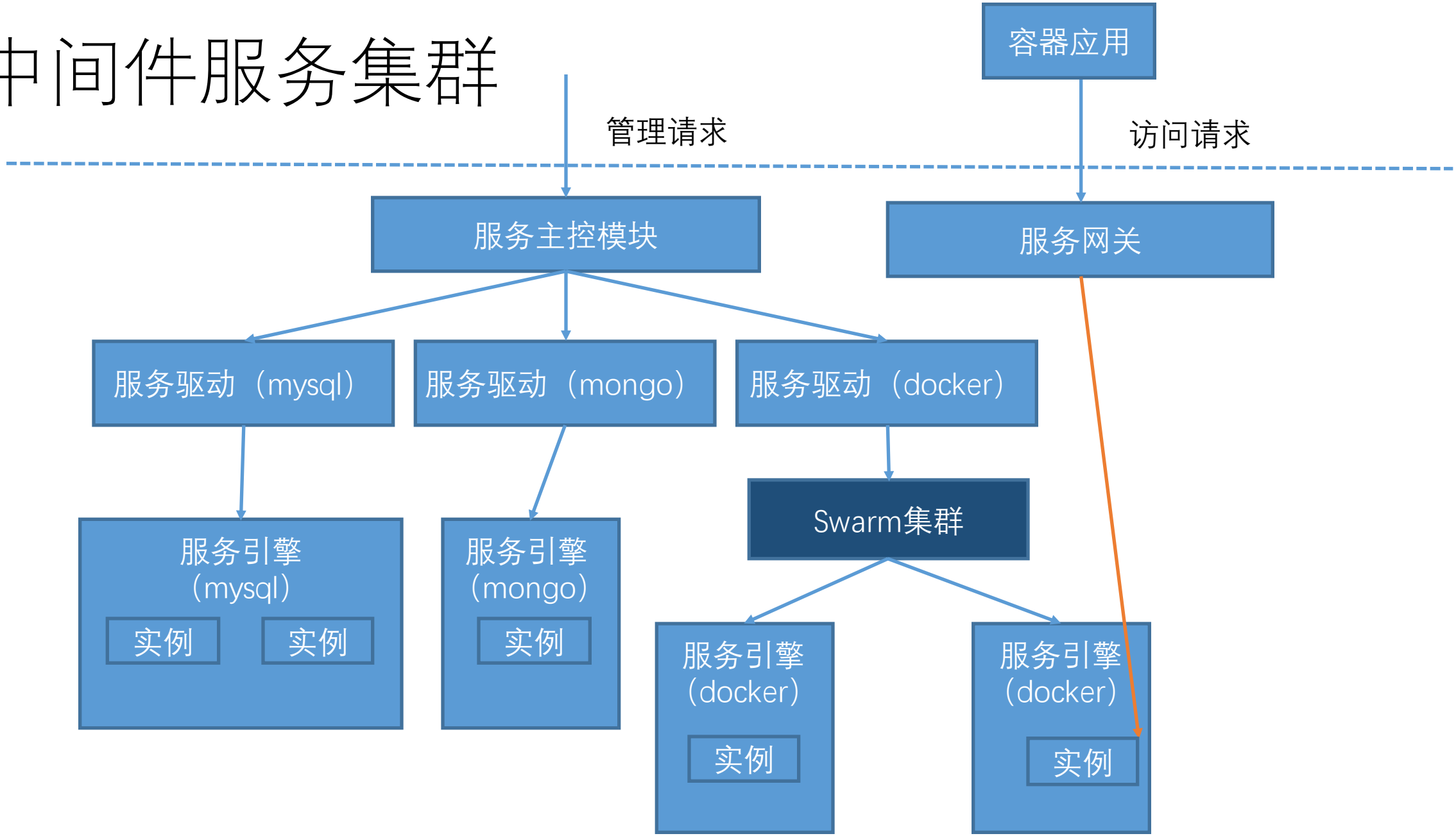
DaoCloud容器云平台



中间件服务集群

- 提供第三方中间件服务
- 接入传统遗留系统服务
- 交付方式：容器与传统形式并存
- 开放性，用户定义服务

中间件服务集群

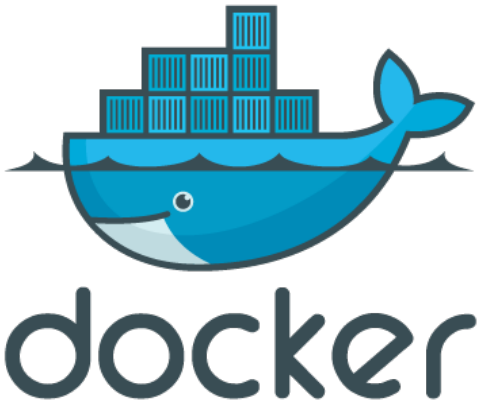

容器应用
管理请求
访问请求
服务主控模块
服务网关
服务驱动（mysql）
服务驱动（mongo）
服务驱动（docker）
服务引擎
（mysql）
实例
实例
服务引擎
（mongo）
实例
Swarm集群
服务引擎
（docker）
实例
服务引擎
（docker）
实例

容器调度本质

- 关键是状态（容器 / 应用）
- 状态的存储（Docker Daemon / ETCD）

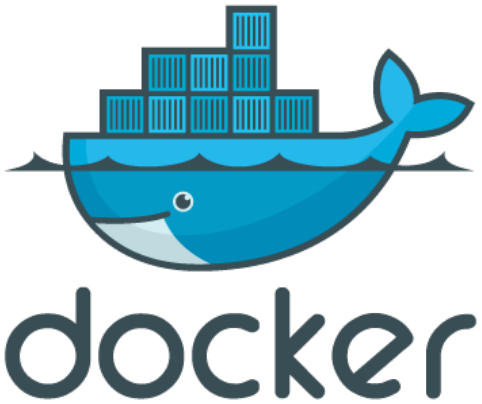
Golang&Docker@DaoCloud

- 微服务架构
- 容器化平台
- Golang in Backends
  - 大部分静态编译
- 容器化粒度部署工具



小结 - Golang&Docker@DaoCloud

- 工程性语言
- 去耦合
- 轻量级



Docker运维&Golang

- 自动化部署工具
  - 容器粒度部署
  - 平台本身的容器化
  - DaoCloud吃狗食
  - 平台的第一次部署

Docker运维&Golang

- Docker管理工具
  - 容器进程数管理
  - 容器Volume的统计与限制
  - 容器特殊状态的汇报
  - 网络带宽限制

Docker运维&Golang

Docker运维经验：
1.尽量减少与Docker Daemon的通信
2.Docker版本的选择
3.可以利用Docker Root
4.建议利用cgroups filesystem

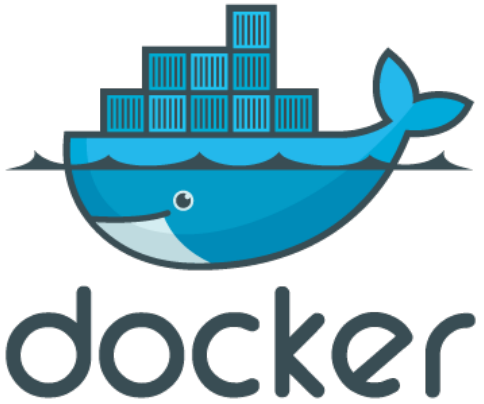
Docker运维&Golang

1.便捷的进程管理

```go
// SetupCgroups applies the cgroup restrictions to the process running in the container based
// on the container's configuration
func SetupCgroups(container *libcontainer.Config, nspid int) (cgroups.ActiveCgroup, error) {
    if container.Cgroups != nil {
        c := container.Cgroups

        if systemd.UseSystemd() {
            return systemd.Apply(c, nspid)
        }

        return fs.Apply(c, nspid)
    }

    return nil, nil
}
```

```go
func (p *setnsProcess) signal(sig os.Signal) error {
    s, ok := sig.(syscall.Signal)
    if !ok {
        return errors.New("os: unsupported signal type")
    }
    return syscall.Kill(p.cmd.Process.Pid, s)
}
```

Docker运维&Golang



2.系统调用 syscall

- 亲和操作系统原语

```
func Bind(fd int, sa Sockaddr) (err error)
func BindToDevice(fd int, device string) (err error)
func BytePtrFromString(s string) (*byte, error)
func ByteSliceFromString(s string) ([]byte, error)
func Chdir(path string) (err error)
func Chmod(path string, mode uint32) (err error)
func Chown(path string, uid int, gid int) (err error)
func Chroot(path string) (err error)
func Clearenv()
func Close(fd int) (err error)
func CloseOnExec(fd int)
func CmsgLen(datalen int) int
func CmsgSpace(datalen int) int
func Connect(fd int, sa Sockaddr) (err error)
func Creat(path string, mode uint32) (fd int, err error)
func DetachLsf(fd int) error
func Dup(oldfd int) (fd int, err error)
func Dup2(oldfd int, newfd int) (err error)
func Dup3(oldfd int, newfd int, flags int) (err error)
```

谢谢