浅谈架构升级

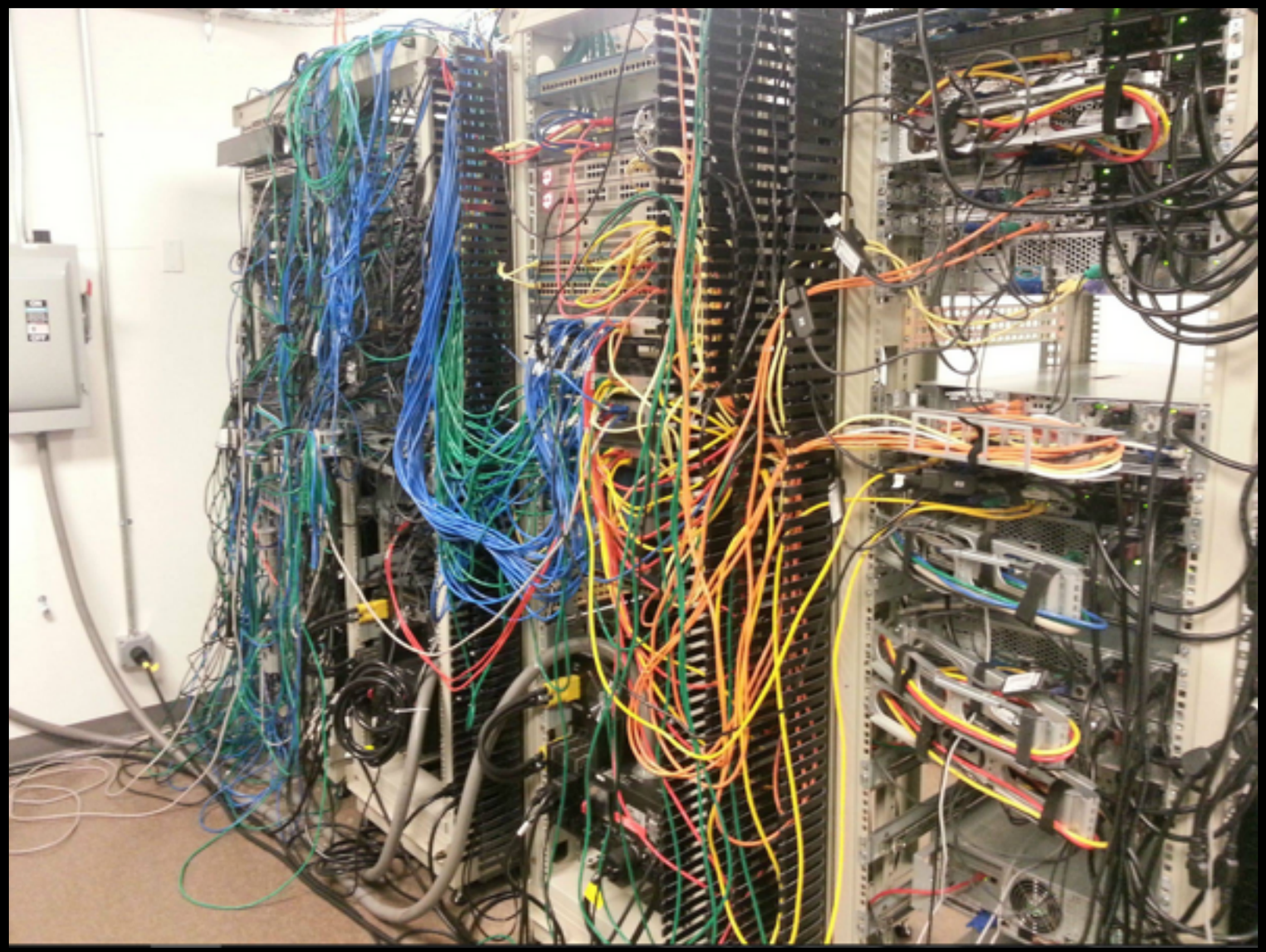
孙宇聪 Coding.net CTO

CODING

CLOUD DEVELOPMENT

CODING @ 2014

CODING @ 2015



CODING @ 2016

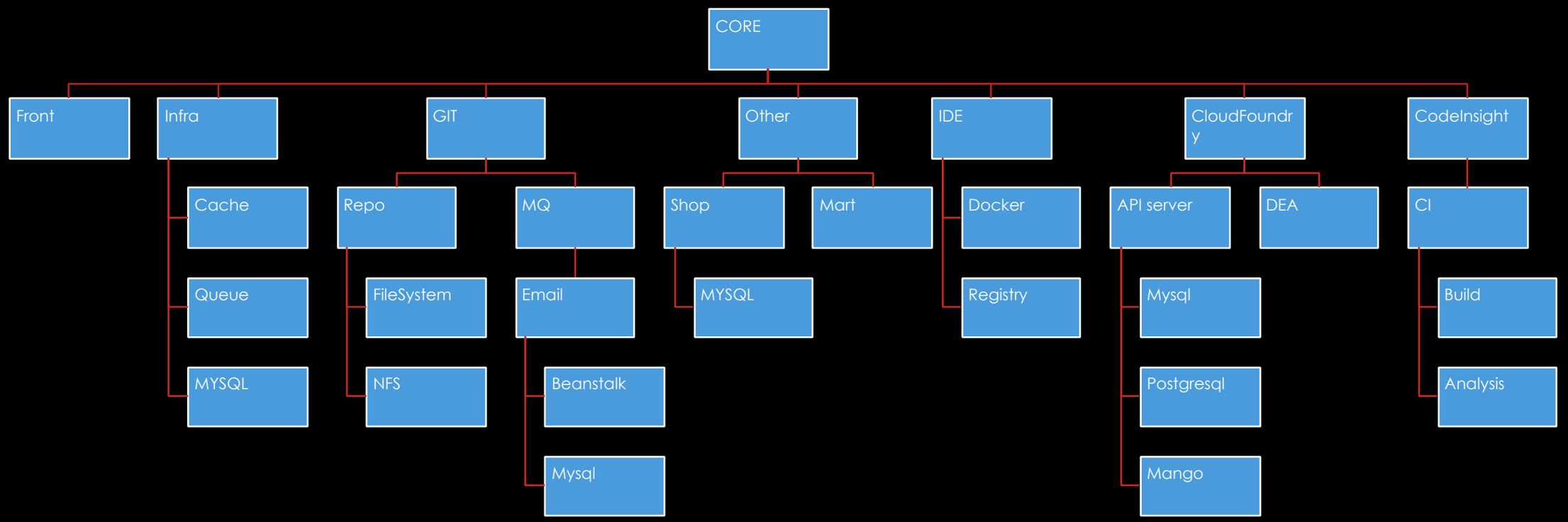
SERIOUSLY, TELL ME

第一个标签 @ 2014-08-06

每周更新至少两次，共计300+个部署标签

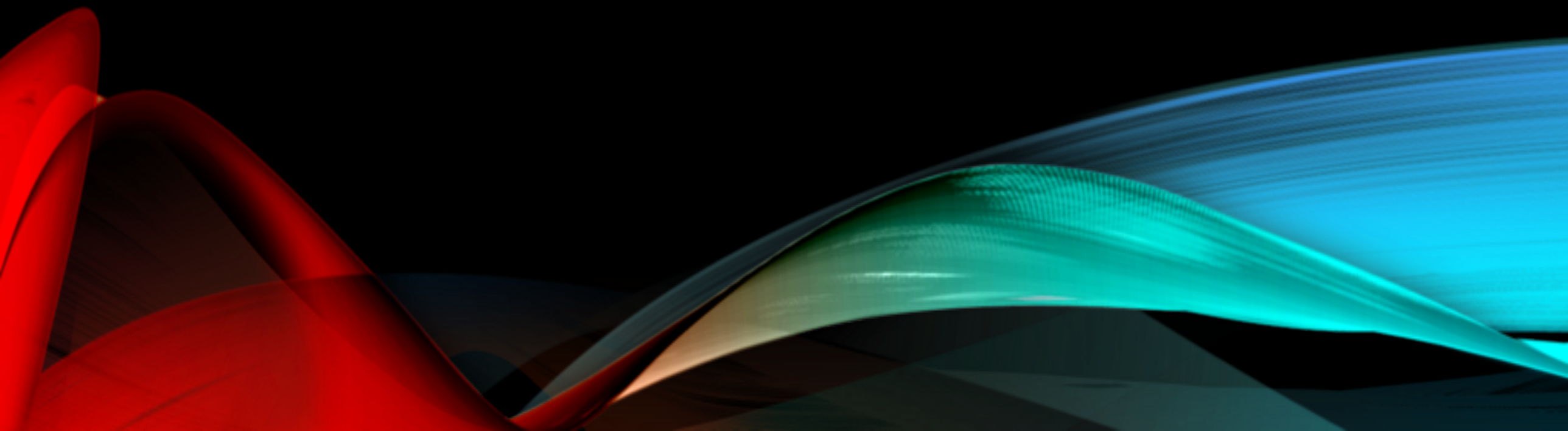
CODING 服务架构 1

CODING 服务架构 2



架构升级就是
飞机飞行过程中更换引擎

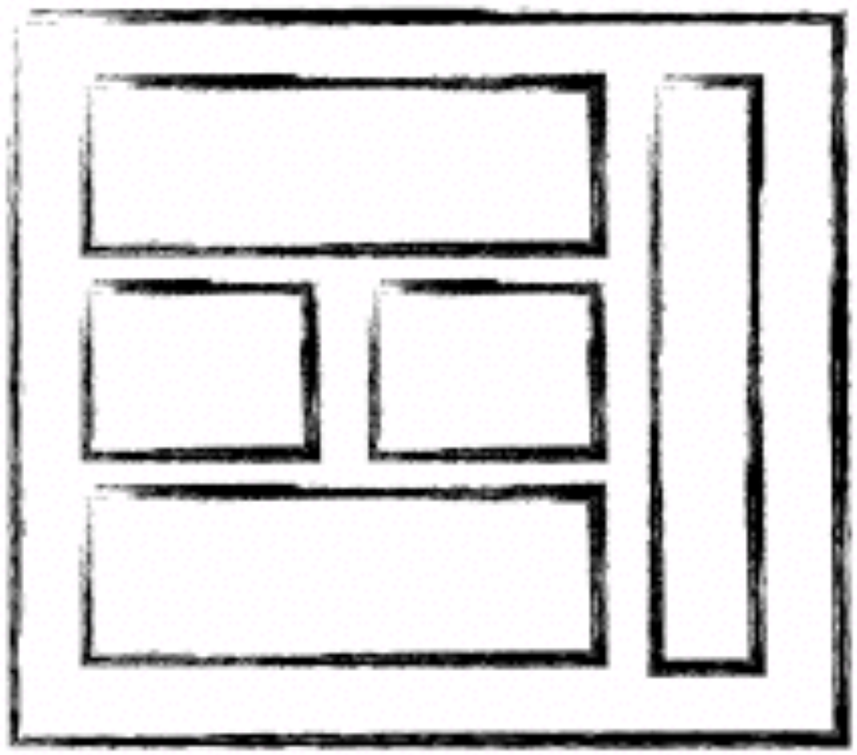
也可以说是给行驶中的火车换轮子。



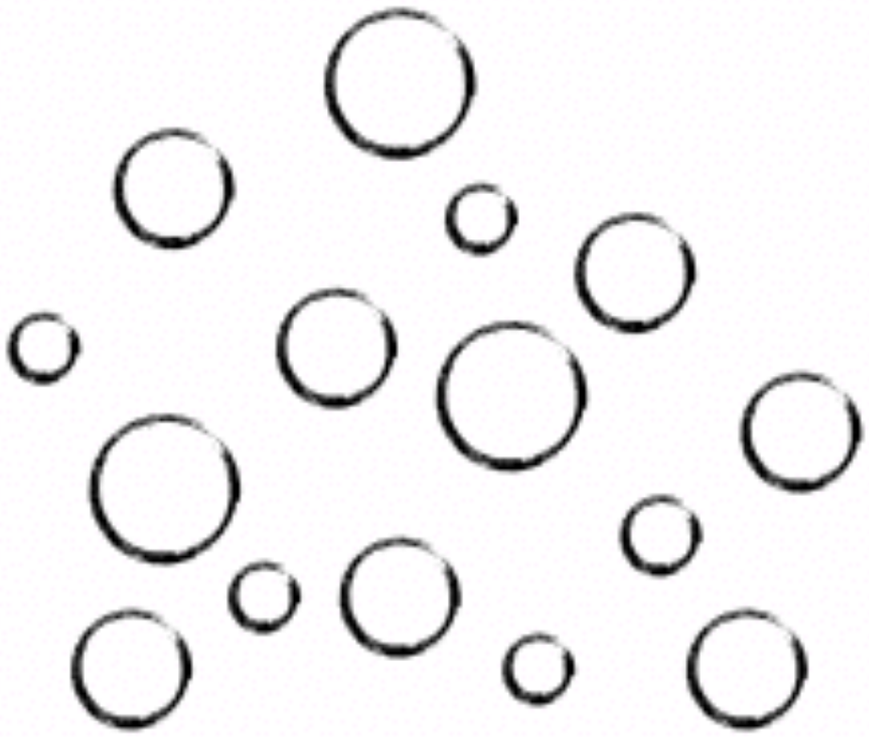
三个关注点

- 研发体系、软件架构的升级
  - 微服务、无状态、数据执行分离
- 生产环境管理理念升级
  - 容器化、代码化、自动化
- 资源管理理念升级
  - Pet Vs Cattle

微服务架构



Monolithic/Layered



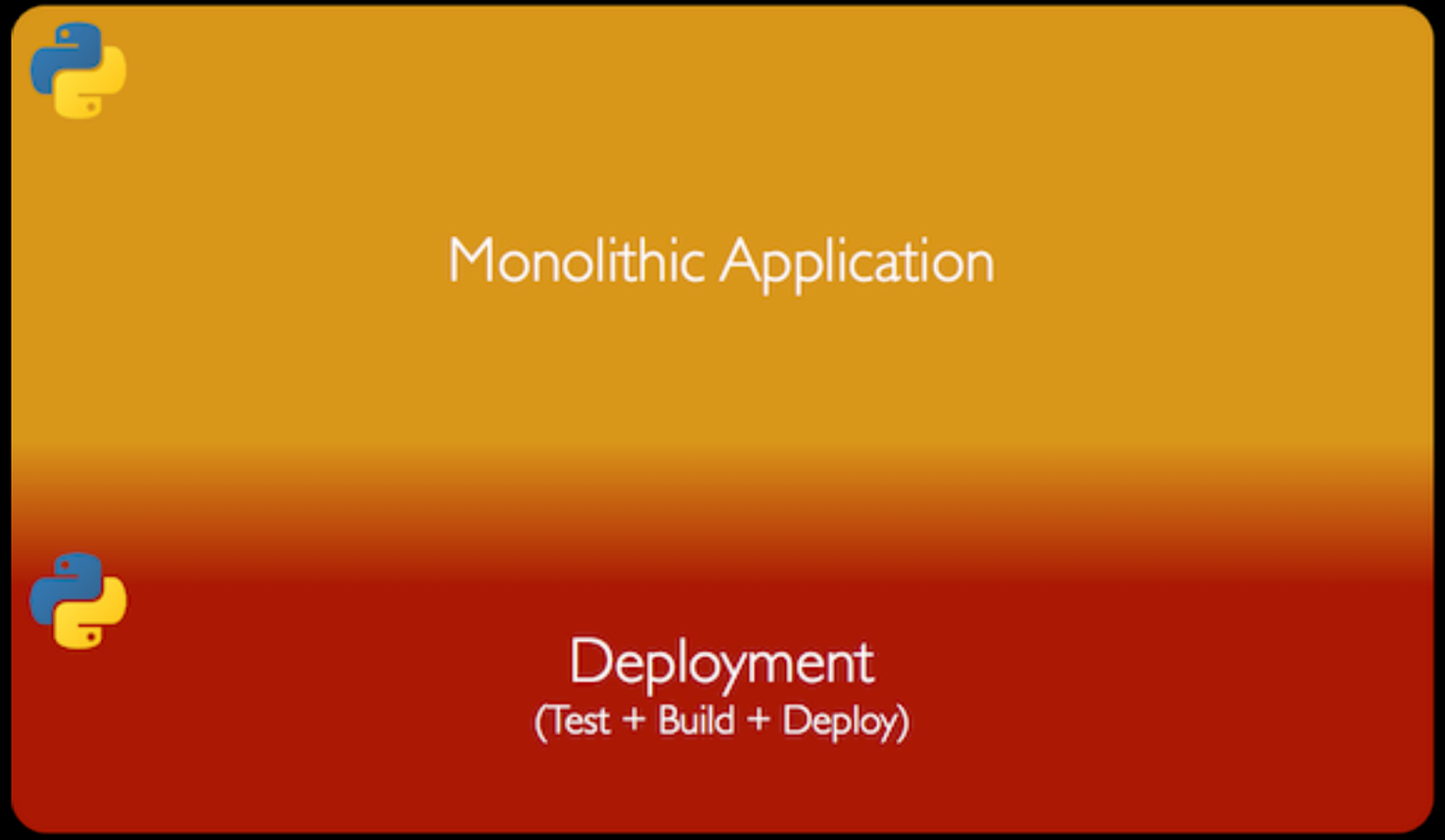
Micro Services

CODING.NET 的成长之路

Feature Team

Feature Team

Feature Team



Ops Team

升级为微服务架构

Feature Team

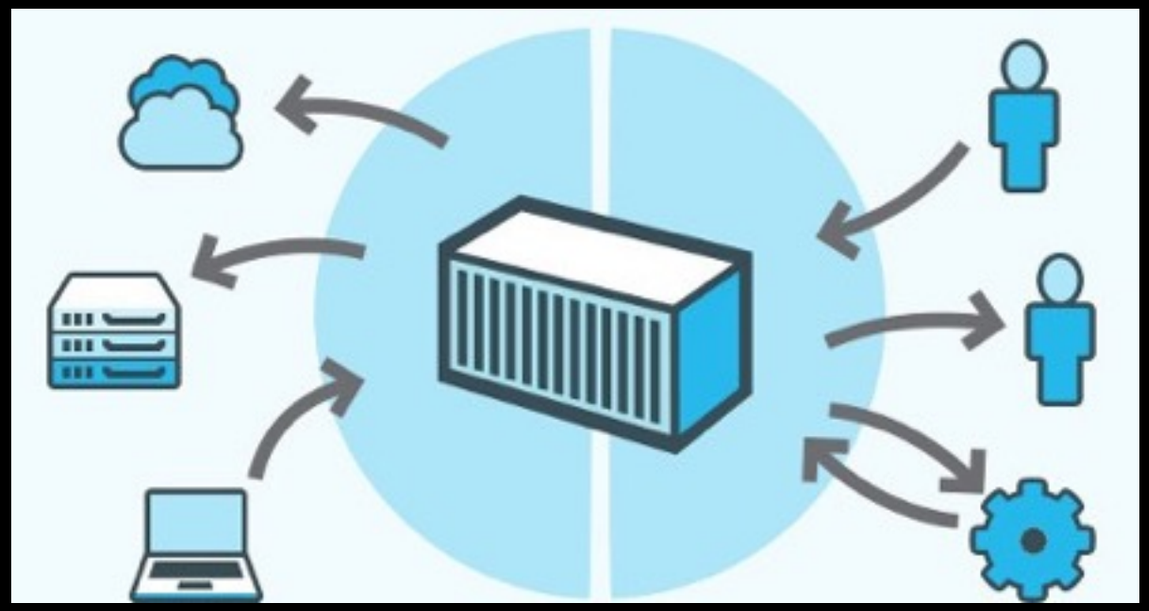
Feature Team

Feature Team

Deployment
(Test + Build + Deploy)

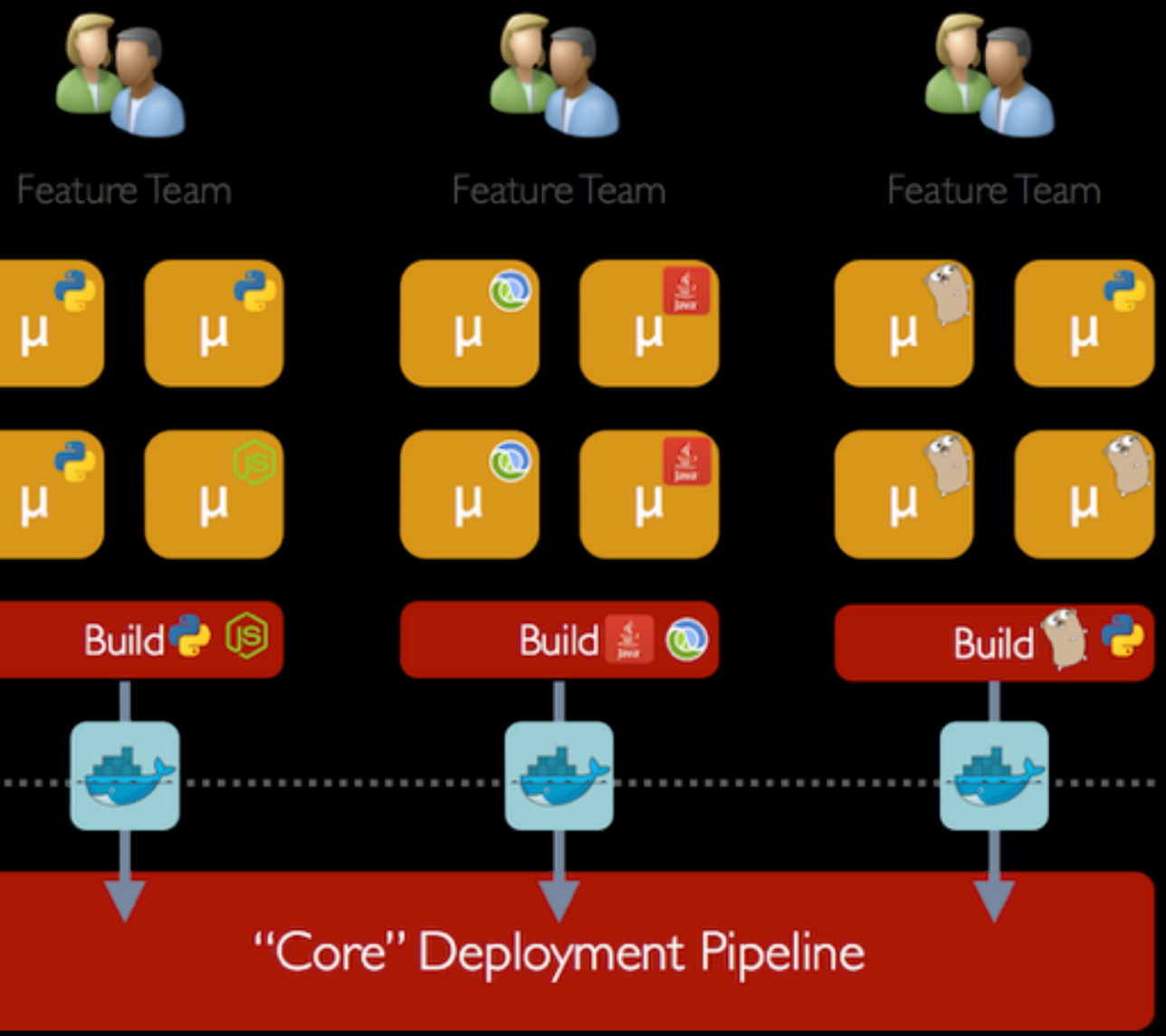
Ops Team

引入容器化技术



- Self Sufficient packaging
- Universal API between Host / App
- Foundation for common infrastructure

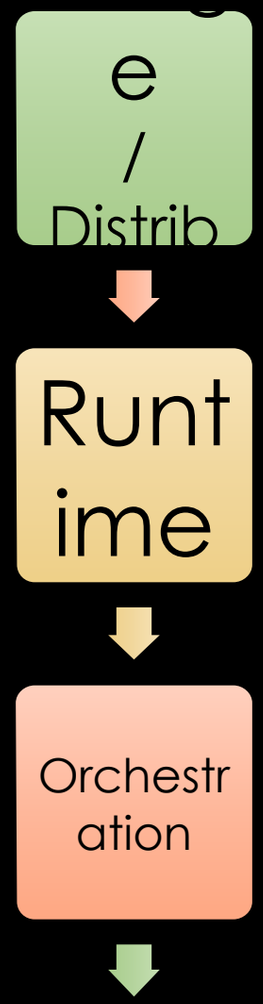
容器化生产环境


Feature Team
Feature Team
Feature Team
Build
Build
Build
"Core" Deployment Pipeline
Ops Team

CODING 与 DOCKER

- Coding 所有代码都打包成了 Docker Image
- Coding 生产环境有自己的私有 Registry
- 用 Docker Container 运行代码
- Coding-job 生产环境容器编排系统

DOCKER 究竟是什么



Docker

鸡肋的 DOCKER IMAGE

一个不那么圆的轮子

DOCKERFILE 有卵用吗

- FROM 的问题
  - 基础镜像没有一个靠谱的
    - Debian, Ubuntu, 还有 Centos/Fedora 基于什么的都有
  - Hub上到处是雷和Abandon-ware
  - 新一代的 copy & paste 大法

DOCKERFILE 有卵用吗

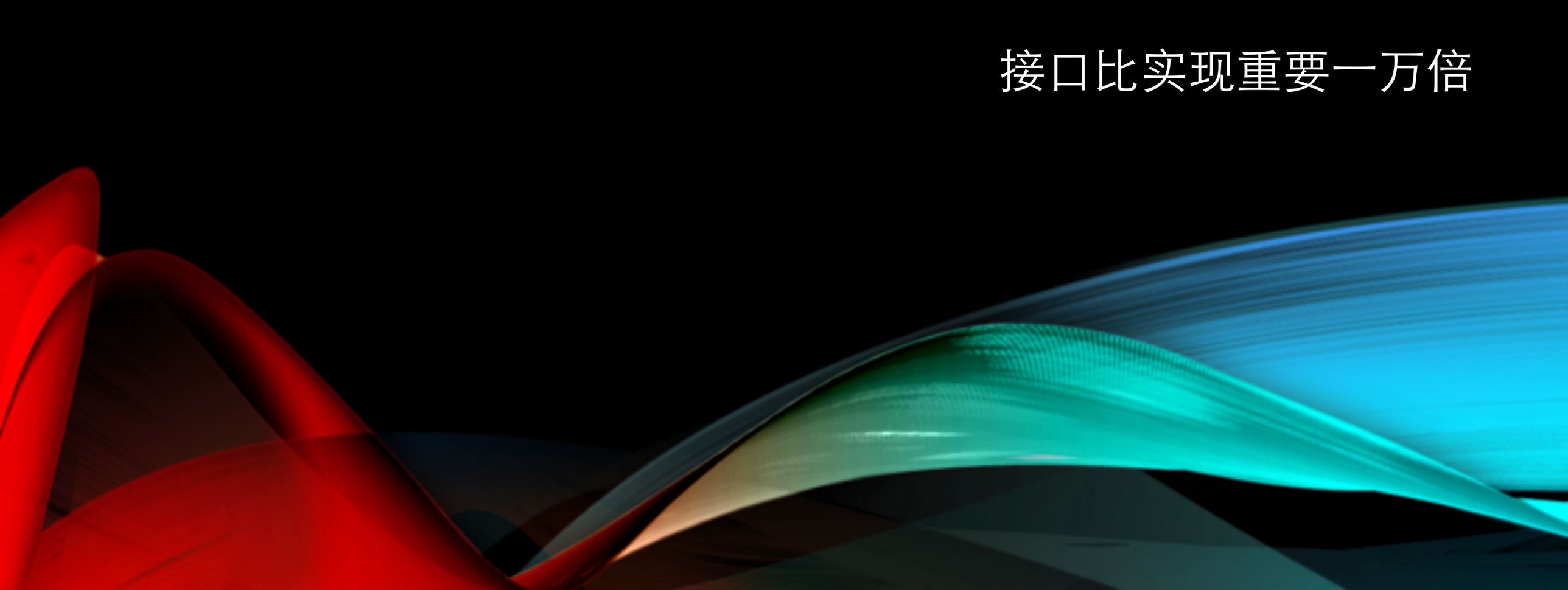
- RUN 的问题
  - apt-get update –y && apt-get upgrade –y
  - 每句加一个layer, 轻松来个十几G的镜像
  - 没完没了的等，继续等，使劲等

DOCKERFILE 有卵用吗

- CMD / ENTRYPOINT 的问题
  - 这俩居然也占layer, 服了.
  - 编译产生的垃圾谁负责? 代码谁负责清理?
  - 有必要把运行命令写死在镜像里吗.

CODING 实践 1: BUILD & PACKAGE

接口比实现重要一万倍



BUILD

- 所有主流编程语言都已然实现了编译和打包工具
  - 大约一万遍

- 依赖管理就不是个问题
  - Vendoring, 自包含镜像，等等，除非姿势不正确。

- 接口统一
  - 实现自己写去

PACKAGE

- 暂时还是先用 Docker Image
  - 随时也可以改

- 正确的 Dockerfile 只有三行
  - FROM base20151030:jre8.20u
  - ADD app.jar /app
  - CMD [“java” , “-jar”, “app.jar”]

- 其实也可以只有两行

废柴的 DOCKER REGISTRY

勉强凑活用，因为实在不重要

REGISTRY 的作用

- Over-engineering 的FTP
- API, 客户端 library 也不那么完善
- 实在排不上优先级去折腾，先凑活用吧

坑爹的 DOCKER RUNTIME

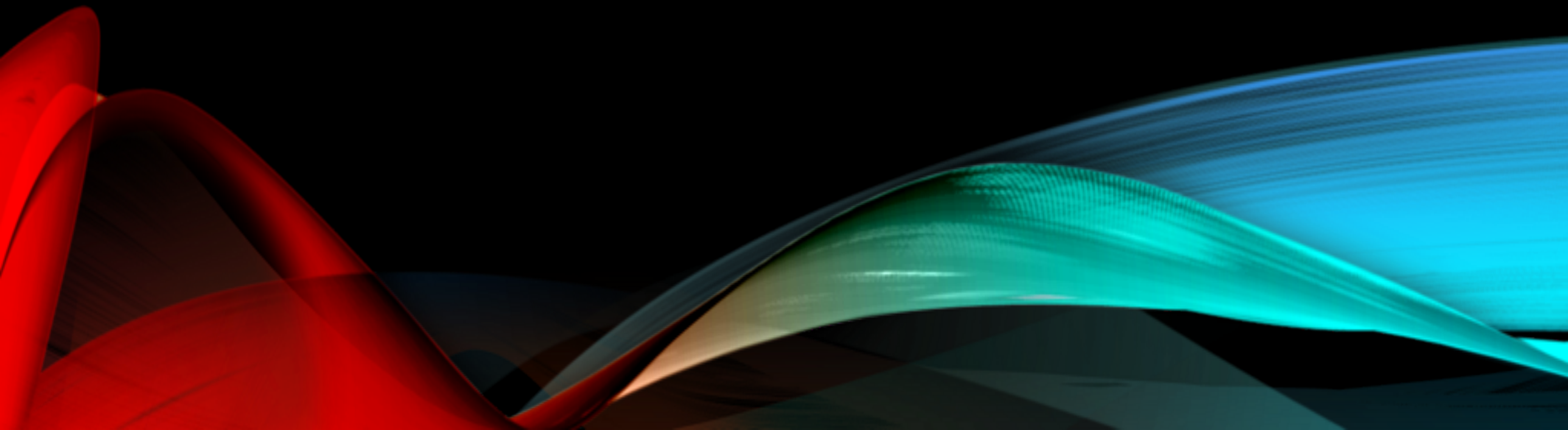
小姐的身子，丫鬟的命

DOCKER DAEMON 大坑实录

- docker container 在 stdout/stderr 有大量数据传输会导致内存泄露，直至 docker daemon OOM
- docker daemon 在频繁创建 container 后，会在 文件系统中遗留很多垃圾文件不清理，导致磁盘 inode 被耗尽
- docker 里面没有init, daemon也没有reap子进程fork很多进程，会在系统中出现很多僵尸进程，最终导致 docker daemon 出现问题。

CODING 实践 2
去其糟粕，取其精华

没有卵用的花哨、前沿功能，一律不用



CODING 的容器配置

- 单进程, 微服务，没必要那么多限制
  - trust-cooperative 环境，我们不是IAAS

- Host 网络模式
  - 不趟SDN, proxy这样的浑水/雷区，性能也不受影响

- host上数据持久化
  - 未来可能可以考虑data container, 但是然并卵
  - 直接 mount Host locale, timezone, passwd 等配置.

不存在的 DOCKER 编排系统

Swarm, K8s, Mesos

神仙打架，百姓遭殃

- Swarm, kubernetes, Mesos
  - 都处于早期，步子迈的太大。

- 动态伸缩的需求有，但是没那么大
  - 在国内的公有云上也是个笑话

- 对 docker 容器的直接接管能力都不行.
  - 需要的功能都没有，没用的烂事一堆

CODING 实践 3
工具、实用、半自动

把有限的精力花在最需要的功能上

代码化的生产环境

```
jobs: <
  name: "coding-prober"
  image: "coding-prober:latest"
  run_on_host: "mon"
  port: "4444"
>
jobs: <
  name: "coding-blog"
  image: "coding-blog:59559fd"
  run_on_host: "core-app-2"
  port: "9090"
>
jobs: <
  name: "webhook-listener"
  image: "webhook-listener:a5af2f2"
  run_on_host: "git-app-1"
  port: "9999"
  envs: <
    key: "http_port"
    value: "8990"
  >
>
jobs: <
  name: "updatehook-listener"
  image: "updatehook-listener:latest"
  run_on_host: "git-app-1"
  port: "9999"
>
```

- 记录什么东西跑在什么机器上
- 常用属性的的管理
- Jobs/Tasks 抽象层
- 集群可复制

半自动化操作

- ./coding-job up coding-mart
  - 自动使用标准配置启动一个容器

- ./coding-job down coding-mart
  - 自动干掉这个容器

- ./coding-job status coding-mart
  - 监控容器的状态，内存等

高级自动化操作

- 执行 update 操作，会列出当前的 image 列表，选择后就可以进行全自动更新。

```
[redacted] go run job.go update coding-blog
Find running container: [/coding-blog_1441006496]
        Version: 59559fd
        Created: Aug 13, 2015 at 11:35:14am
        ImageID: e7265fa09e1c37df11701112deca2f8a305905f745e6aa4ec3e749ecd1fec438
        ContainerID: a00c4832d275068ced5b96bbffe631df01dad3378613e4b29b9646311e417545
The tags in the registry
        Version: 2b19fca        IMAGE ID: 768454c913517f5dfdb77e1a9f90c296921fa81af406c206f668d41f1af9ba8c      CREATED:
 Sep 11, 2015 at 2:26:02pm
        Version: latest         IMAGE ID: 768454c913517f5dfdb77e1a9f90c296921fa81af406c206f668d41f1af9ba8c      CREATED:
 Sep 11, 2015 at 2:26:02pm
        Version: 59559fd        IMAGE ID: e7265fa09e1c37df11701112deca2f8a305905f745e6aa4ec3e749ecd1fec438      CREATED:
 Aug 13, 2015 at 11:35:14am
        Version: 0666880        IMAGE ID: 3e7ecba08d924e3704bb6f0d79a339f12be9aa548c202521175b9d9a5c41dbfc      CREATED:
 Aug 6, 2015 at 11:19:16am
        Version: a0a97e3        IMAGE ID: a5f4613a4c146151a52ccd7d2208bc8352b7b0aa59f27493b5ccbcf9c1c0b80c      CREATED:
 Aug 6, 2015 at 10:53:30am
        Version: e2c6ffc        IMAGE ID: a1e203726403e963c4fe9867ca4eae7807d8932f92ba036457e60e859e7bbf17      CREATED:
 Jul 31, 2015 at 11:21:21am
        Version: 0426a0c        IMAGE ID: cfcfc4c6fb782af37479fbc27f4e1b45deb3fcc148533093d3f8db519cd913b6      CREATED:
 Jul 28, 2015 at 10:57:28pm
        Version: 7363309        IMAGE ID: 6dfd077e05841d34b45cc3d62552d1ff9127ad1784fd482471194717d0f38008      CREATED:
 Jul 28, 2015 at 9:42:41pm
        Version: fd89efd        IMAGE ID: a88a6071250d7a363895af59f3fd681d4d32b2073b7317570f32347905f67f40      CREATED:
 May 27, 2015 at 10:30:15am
You can run 'go run job.go update jobname_tag'.
```

未来还准备做的功能

- ./coding-job diff
  - 在真的更新之前，先看看改了什么

- ./coding-job rolling-update
  - Job/Tasks 抽象层可以帮助很快的批量操作一个服务

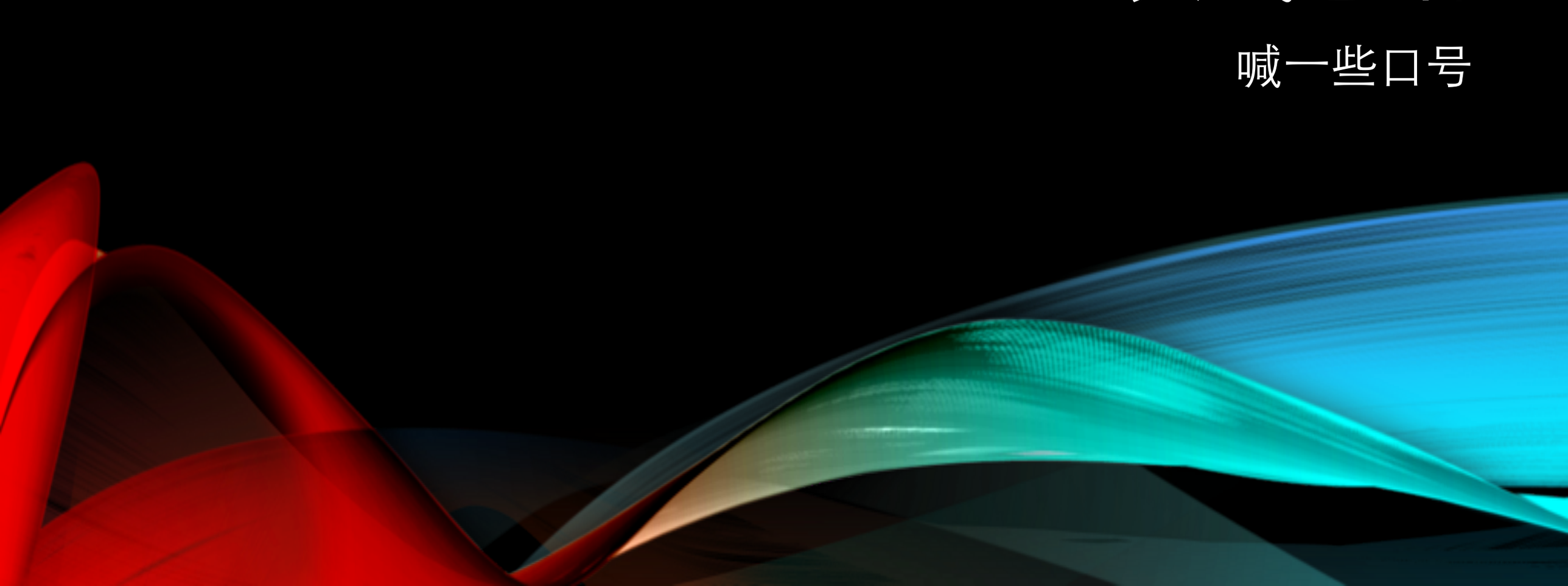
- ./coding-job web
  - 方便网页化, 可以看 status, log, 甚至远程 shell 功能

CODING 生产环境管理

- Pet vs Cattle
  - 重装一台机器要多久?
- 静态，手动的资源调配
  - 多留点富余量，迁移能力比压榨能力更重要
- Redundancy 更重要
  - 要做杀不死的小强

CODING 实践总结

喊一些口号



开发环境：定死接口，放手实现

生产环境容器化：关注三个重点

打成包

挪的动

可复制

DEVOPS 切身实践

Q & A

BUILD → PACKAGE → RUN → 打成包