ThoughtWorks® IT大咖说 知识共享平台

容器的编排和部署

2018.1.28 刘祥旭

应用服务架构演进

单体应用 → N-层架构 → 微服务架构

物理服务器

虚拟化

容器平台

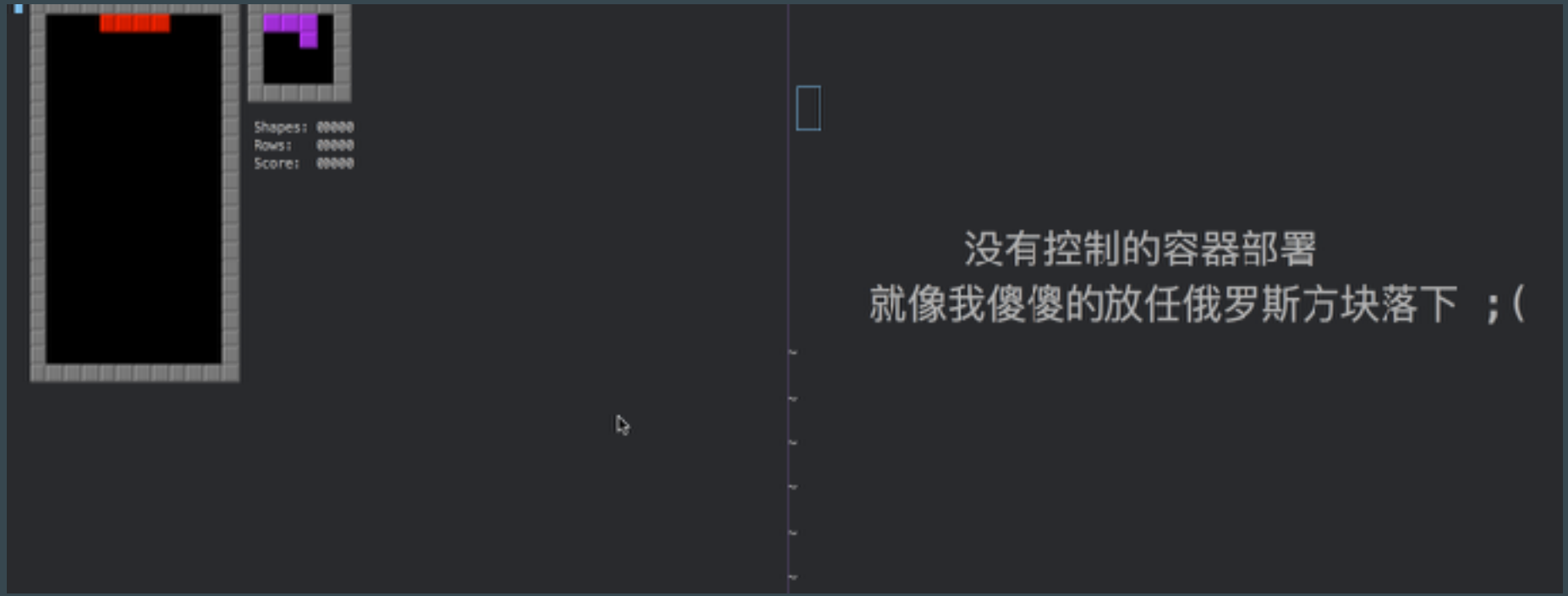
低成本，易维护，灵活，越来越快速的把应用推向市场

应用的维护模式

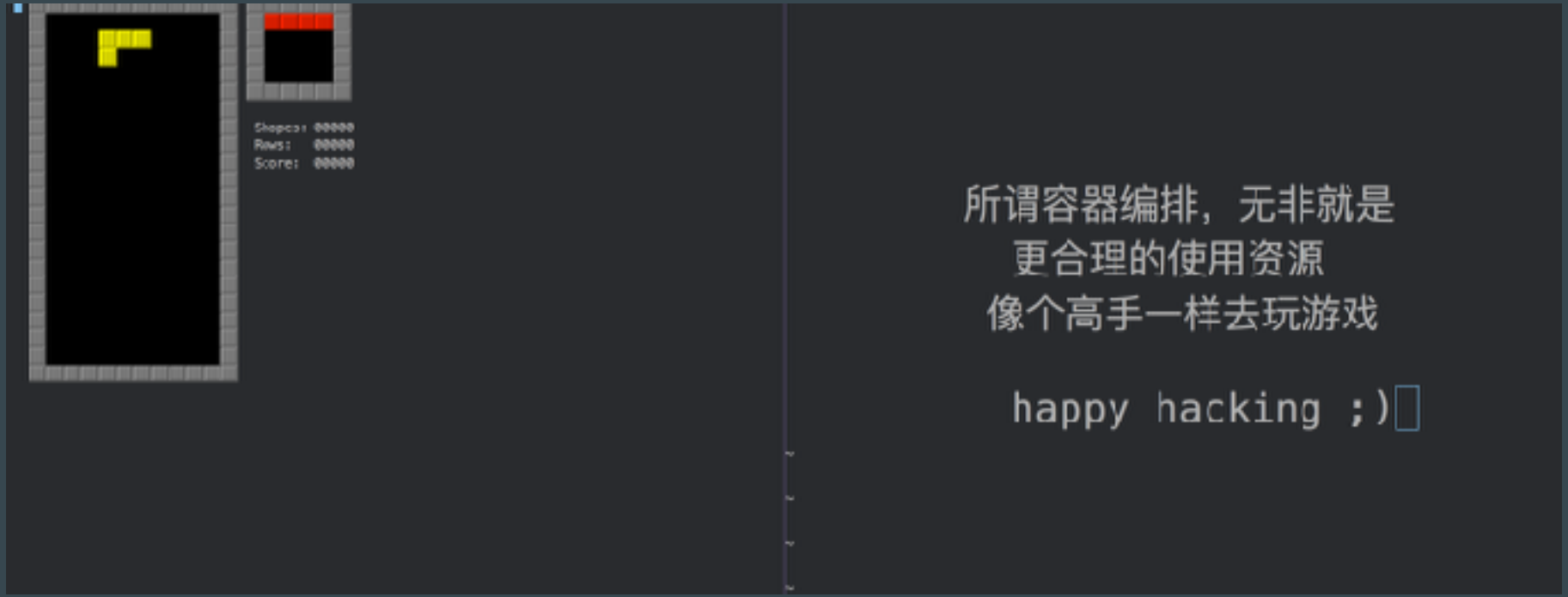
宠物模式

围栏模式

容器编排


Shapes: 00000
Rows: 00000
Score: 00000
没有控制的容器部署
就像我傻傻的放任俄罗斯方块落下 ;(

容器编排


Shapes: 00000
Rows: 00000
Score: 00000
所谓容器编排，无非就是
更合理的使用资源
像个高手一样去玩游戏
happy hacking ;)

主流的容器编排引擎

Swarm

Docker公司原生容器编排工具

- 内置在Docker引擎中
- 负载均衡
- 服务发现
- Docker原生命令行和API
- 高级网络功能
  - Mesh network
  - vxlan

Kubernetes

Google多年大规模容器管理技术的开源版本

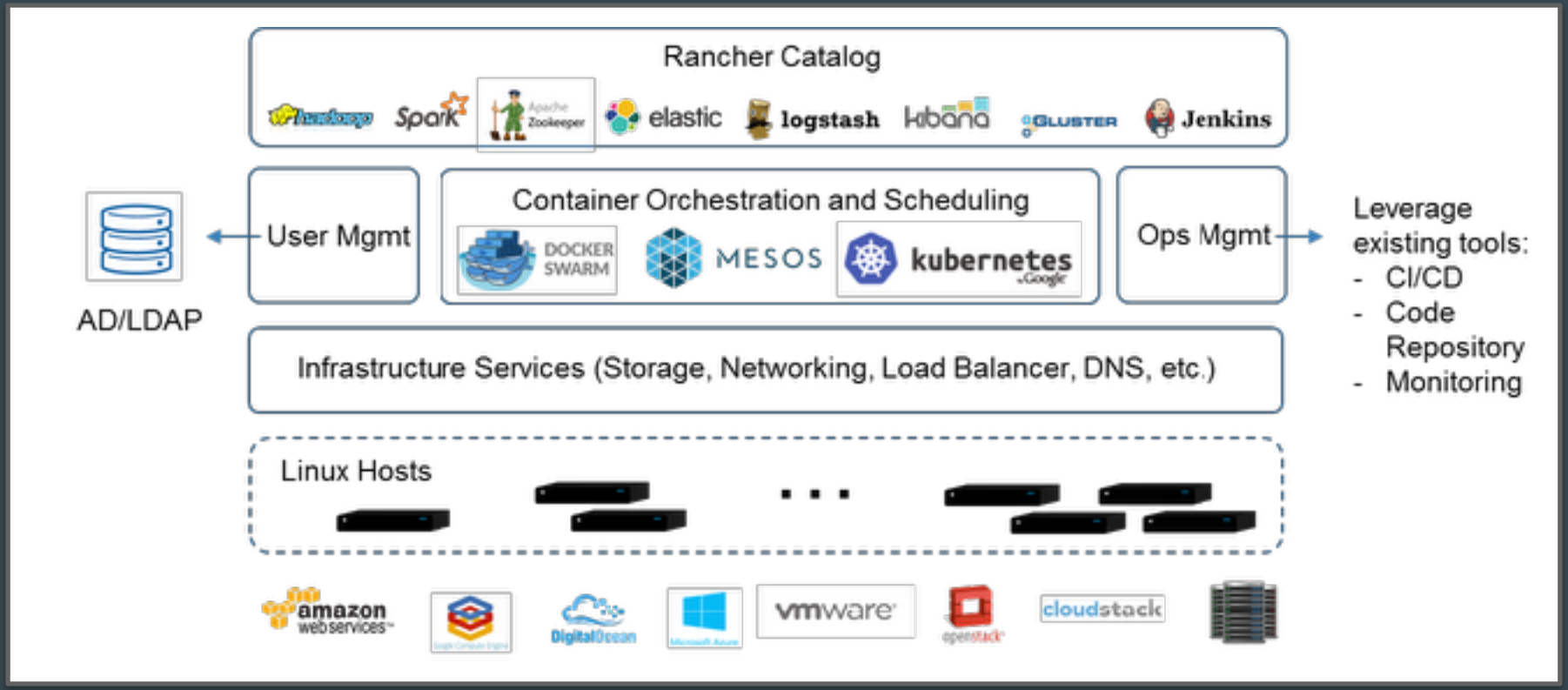
- 丰富的概念模型
- 大规模的生态系统
- 灵活的标签和选择器
- 网络功能
  - 负载均衡
  - 服务发现

Rancher

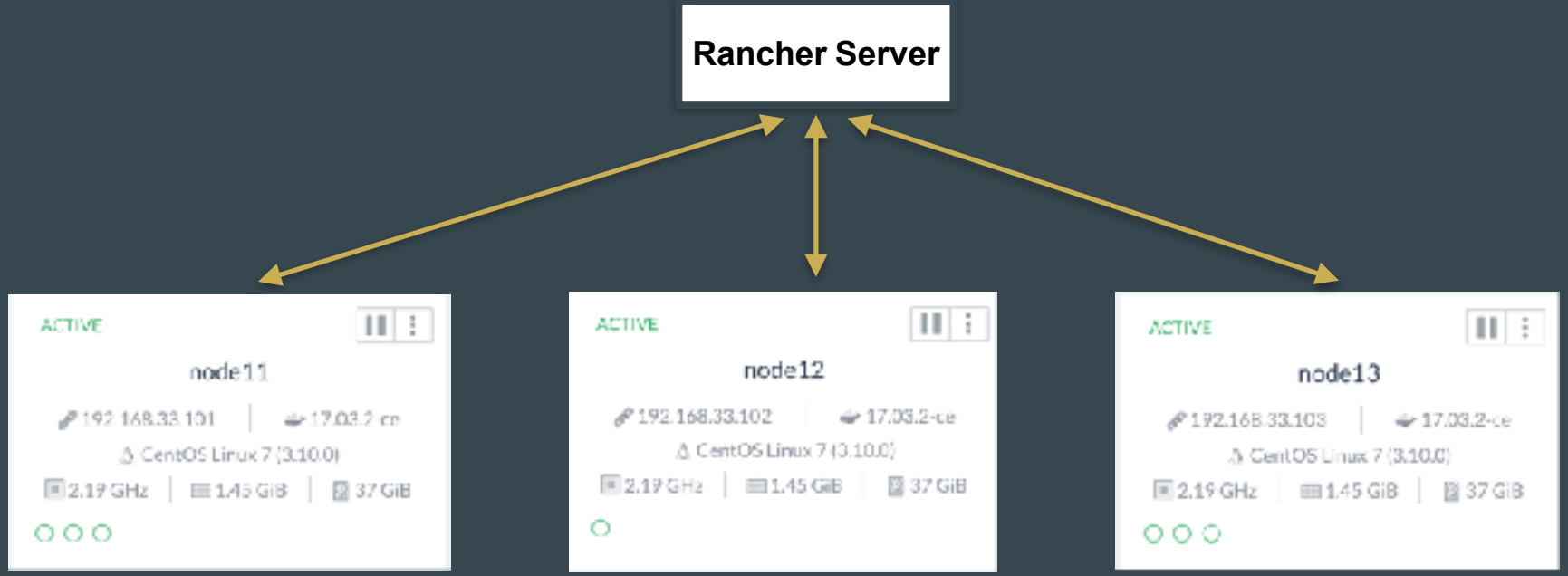
开源的企业级全栈化容器部署及管理平台

- UI界面易用，部署简单
- 支持多种编排引擎
- 应用服务目录

Rancher架构介绍



Rancher Demo环境

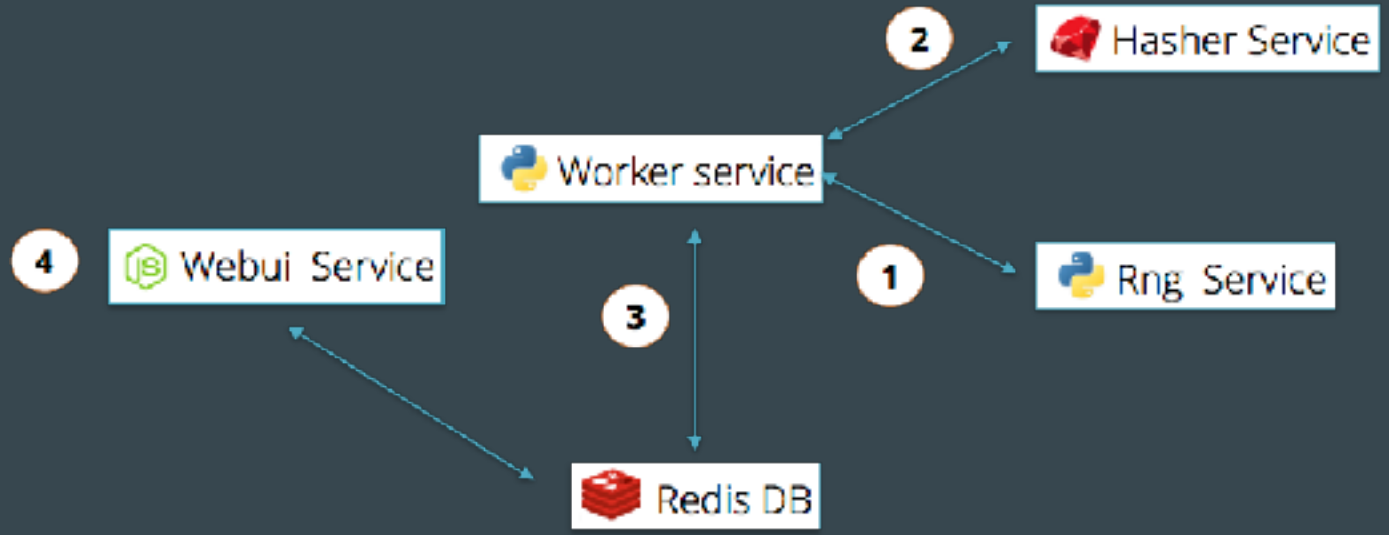


Demo应用

Docker币

https://github.com/jpetazzo/orchestration-workshop

Docker币应用介绍



1. Worker向Rng请求随机bytes数据
2. Worker将bytes数据发送至hasher，计算返回hash值
3. Worker判断hash值以0开始，则认为挖一个Docker币，把hash值存入redis数据库
4. Webui用于展示每秒产生的docker币数量

1.构建容器镜像

构建容器镜像

- 使用最小镜像 alpine
- 一个容器一个进程
- 日志输出到stdout
- 多stage构建

2.编排服务文件docker-compose

编写docker-compose文件

```yaml
version: "2"
services:
  rng:
    image: 192.168.33.101:5000/rng:v0.1
    ports:
    - "8001:80"
  hasher:
    image: 192.168.33.101:5000/hasher:v0.1
    ports:
    - "8002:80"
  webui:
    image: 192.168.33.101:5000/webui:v0.1
    ports:
    - "8000:80"
  redis:
    image: redis
  worker:
    image: 192.168.33.101:5000/worker:v0.01
```

3. 使用rancher命令行部署

部署到rancher容器平台

- 在rancher上生成管理key，执行部署的服务器或CI服务器上
- 使用rancher-compose 命令部署

```
# 在rancher界面生成管理KEYS
$ export RANCHER_URL=http://<server_ip>:8080
$ export
RANCHER_ACCESS_KEY=<accessKey_of_account_api_key>
$ export RANCHER_SECRET_KEY=<secretKey_of_account_api_key>

# 开始部署

$ rancher-compose -p dockercoins up -d
```

4.Scale up应用

```
$ rancher-compose -p <stack_name> scale <service_name>=3
```

5. Rolling Upgrade

```
$ rancher-compose -p <stack_name> \
      upgrade <service_name> <service_name1>
```

一些实践

- Labels 调度
- 资源限制
- 服务健康检查

容器编排-调度

- Lables调度

```
rng:
  image: 192.168.33.101:5000/rng:v0.1
  labels:
    io.rancher.scheduler.affinity:container_label_ne: io.rancher.stack_service.name=$${stack_name}/$${service_name}
```

- 资源限制调度

```
hasher:
  mem_limit: 104857600
  image: 192.168.33.101:5000/hasher:v0.1
  mem_reservation: 52428800
```

容器编排-健康检查

- 服务健康检查

```
version: '2'
services:
  webui:
    scale: 1
    start_on_create: true
  rng:
    scale: 3
    start_on_create: true
    health_check:
      healthy_threshold: 2
      response_timeout: 2000
      port: 80
      unhealthy_threshold: 3
      initializing_timeout: 60000
      interval: 2000
      strategy: recreate
      reinitializing_timeout: 60000
```

Q&A

Thank you