

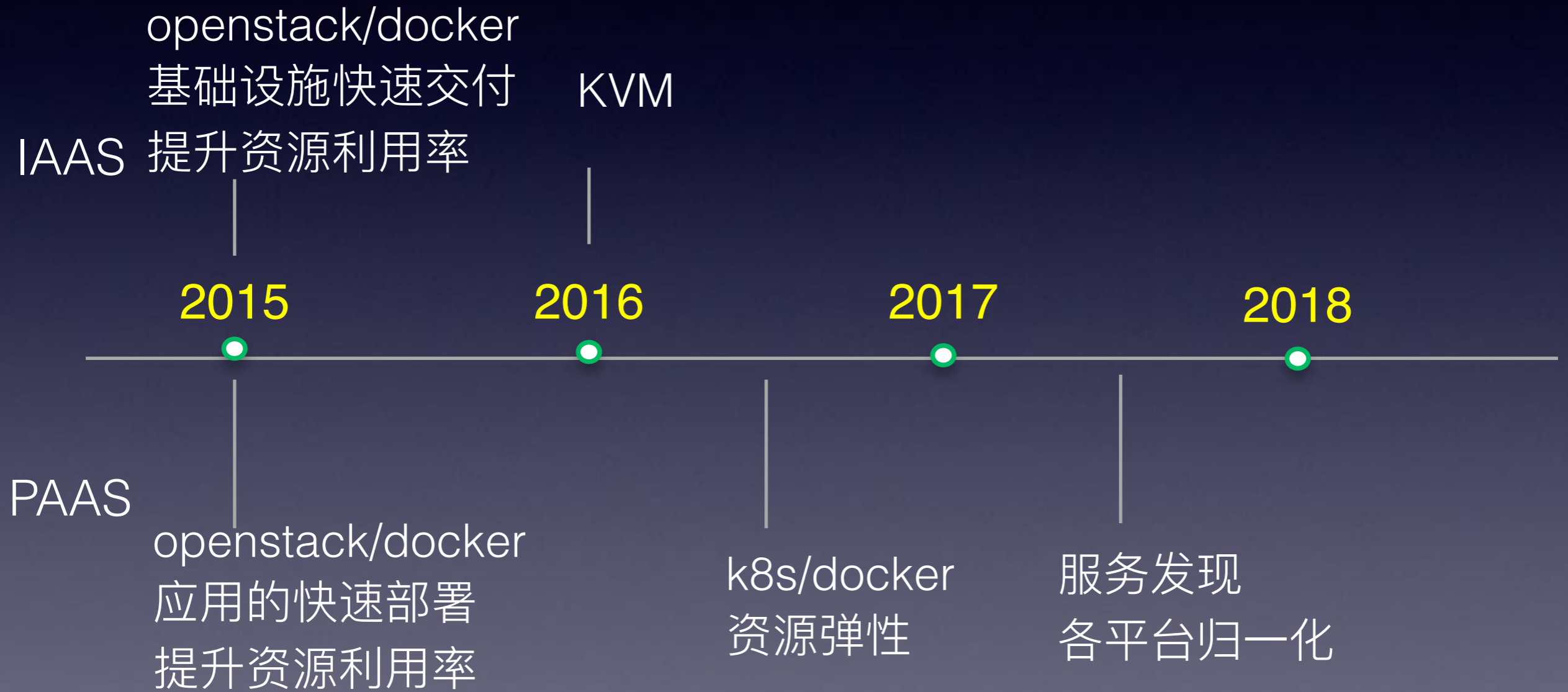
# 蘑菇街的PAAS平台实践

牧白 (王锐)

# 提纲

- 技术架构
- 特性讲解
- 实践案例
- 规划

# 演进



# 技术架构



# 镜像管理

- 需求

集群间镜像同步

权限认证

屏蔽dockerfile

其他管理上的需求： 查找镜像最新的tag

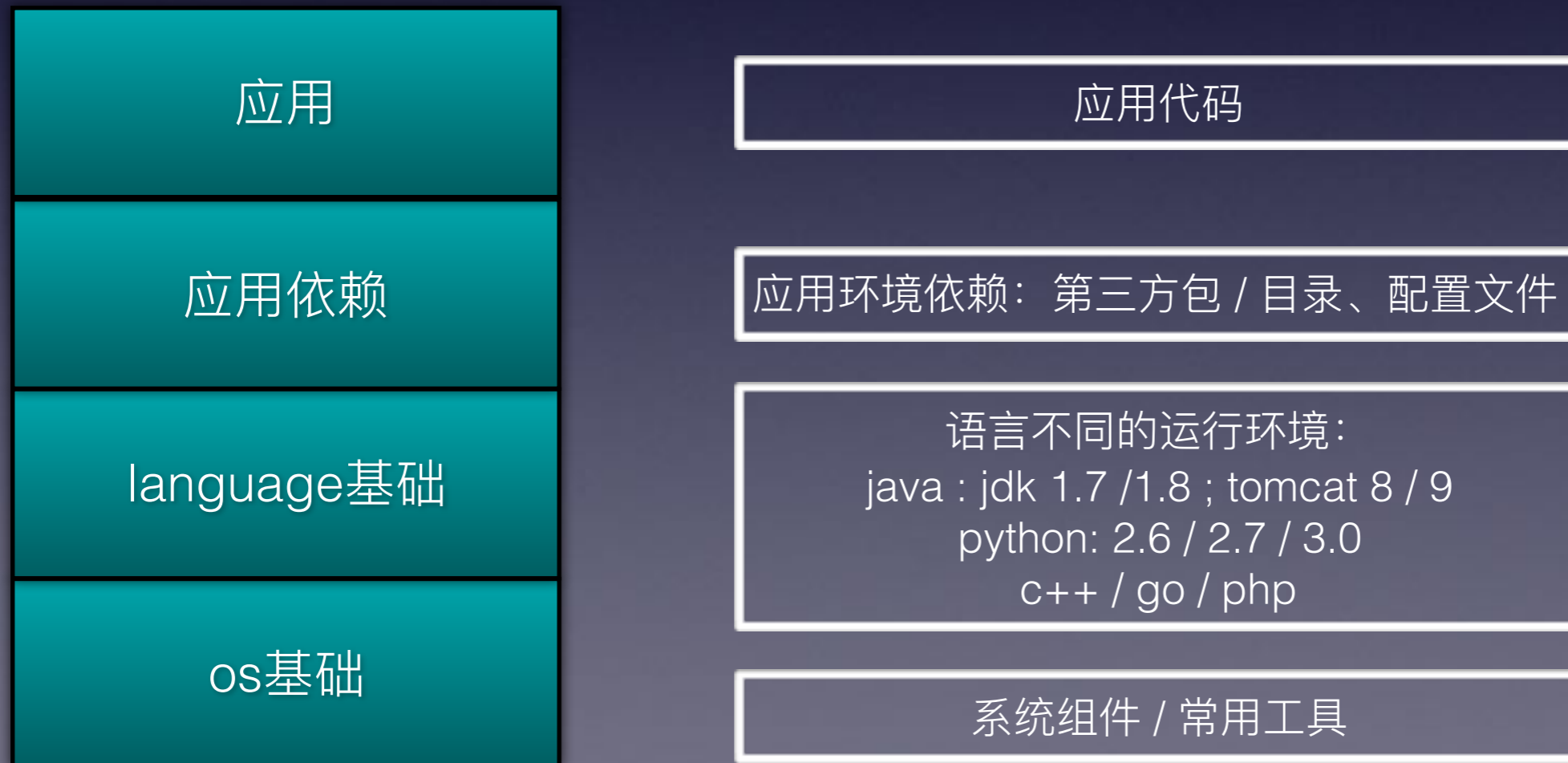
- Why not Harbor ?

# 镜像管理—技术架构



# 镜像管理—分层

- 逻辑上分四层



# 镜像管理—分层

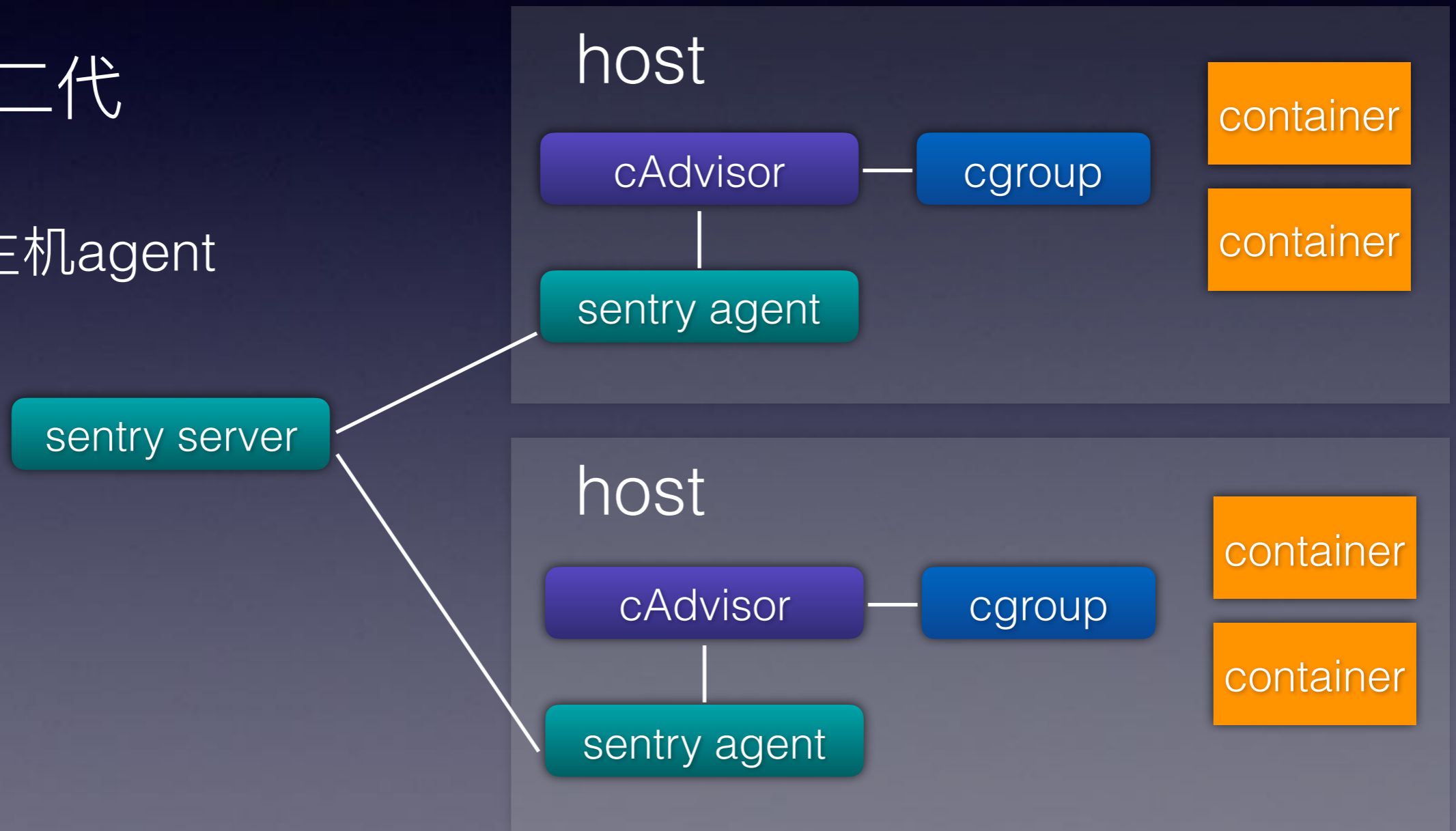
- 经验总结
  - 不变的放下层，易变的放上层
  - 尽量复用下层，下层变更后，有流程保障更新
  - 适当利用 dockerfile 中的 ONBUILD 钩子
  - 如果 dockerfile 对外开放，要严格把关





# 监控

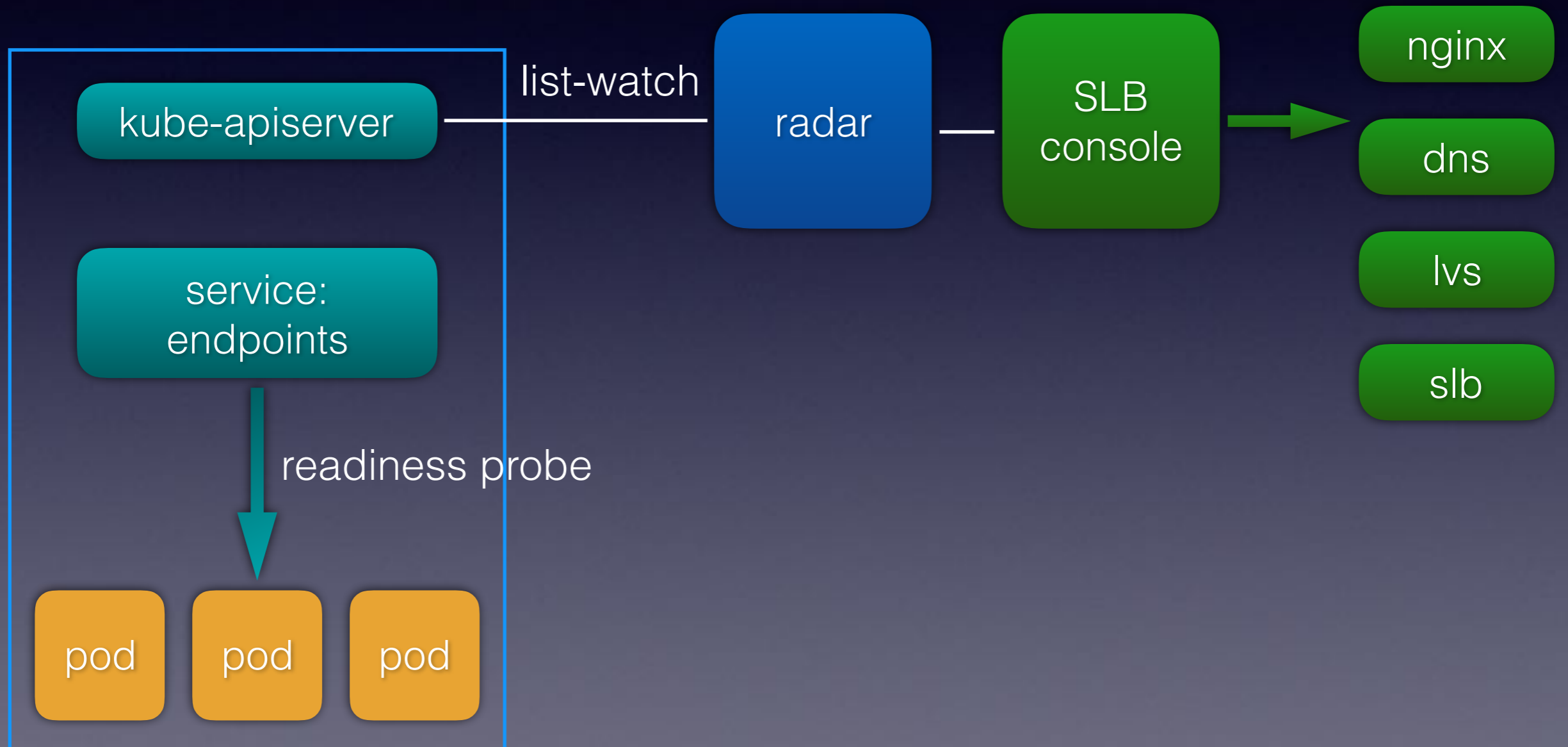
- 第二代  
宿主机agent



# 监控

- 监控项加强
  - 连接跟踪数: `nf_conntrack_count / nf_conntrack_max`
  - 进程使用数和上限: `kernel.pid_max`
  - docker thin pool 使用率
  - 容器 oom 监控: `cgroup memory.oom_control`
  - pod evict 监控
  - 参考了 prometheus , 将 k8s 集群状态纳入了监控

# 服务发现



# 服务发现

- 为何没全部使用开源
- 为何使用headless service
- 健康检查支持 4层/7层/CMD
- nginx改造
- 主动的上下线

# 实践—短时任务

- 特点
  - 任务跑完即可释放资源
  - 持续时间几秒钟到几小时不等
  - 任务是离线的，即便失败不会直接影响线上业务
  - 往往在凌晨执行

# 实践—短时任务

- 支持的策略
  - 调度：即时、指定时刻、cron定时
  - 同步 / 异步
  - 回调
  - 超时
  - 失败后保留实例

# 实践—短时任务

- 集群
- 事件
- 镜像仓库
- 机器池
- 节点管理
- 短时任务
- 用户管理

短时任务基地

添加短时任务

请输入任务ID



刷新

任务ID	创建时间	更新时间	主机IP	资源是否已回收	状态	操作
338192	2018-3-12 19:49:51	2018-3-12 19:50:15	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338191	2018-3-12 19:49:9	2018-3-12 19:49:36	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338190	2018-3-12 19:48:26	2018-3-12 19:49:30	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338189	2018-3-12 19:47:4	2018-3-12 19:48:21	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338188	2018-3-12 19:45:52	2018-3-12 19:46:12	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338187	2018-3-12 19:45:11	2018-3-12 19:45:16	[REDACTED]	未回收	running	<a href="#">详情</a> <a href="#">删除</a>
338186	2018-3-12 19:44:28	2018-3-12 19:44:52	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338185	2018-3-12 19:42:17	2018-3-12 19:43:9	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>
338184	2018-3-12 19:41:35	2018-3-12 19:41:56	[REDACTED]	已回收	success	<a href="#">详情</a> <a href="#">删除</a>



# 实践—短时任务

- 效果
  - 资源高度复用、弹性
  - 自动化部署和收集结果，提升效率

# 实践——调度策略

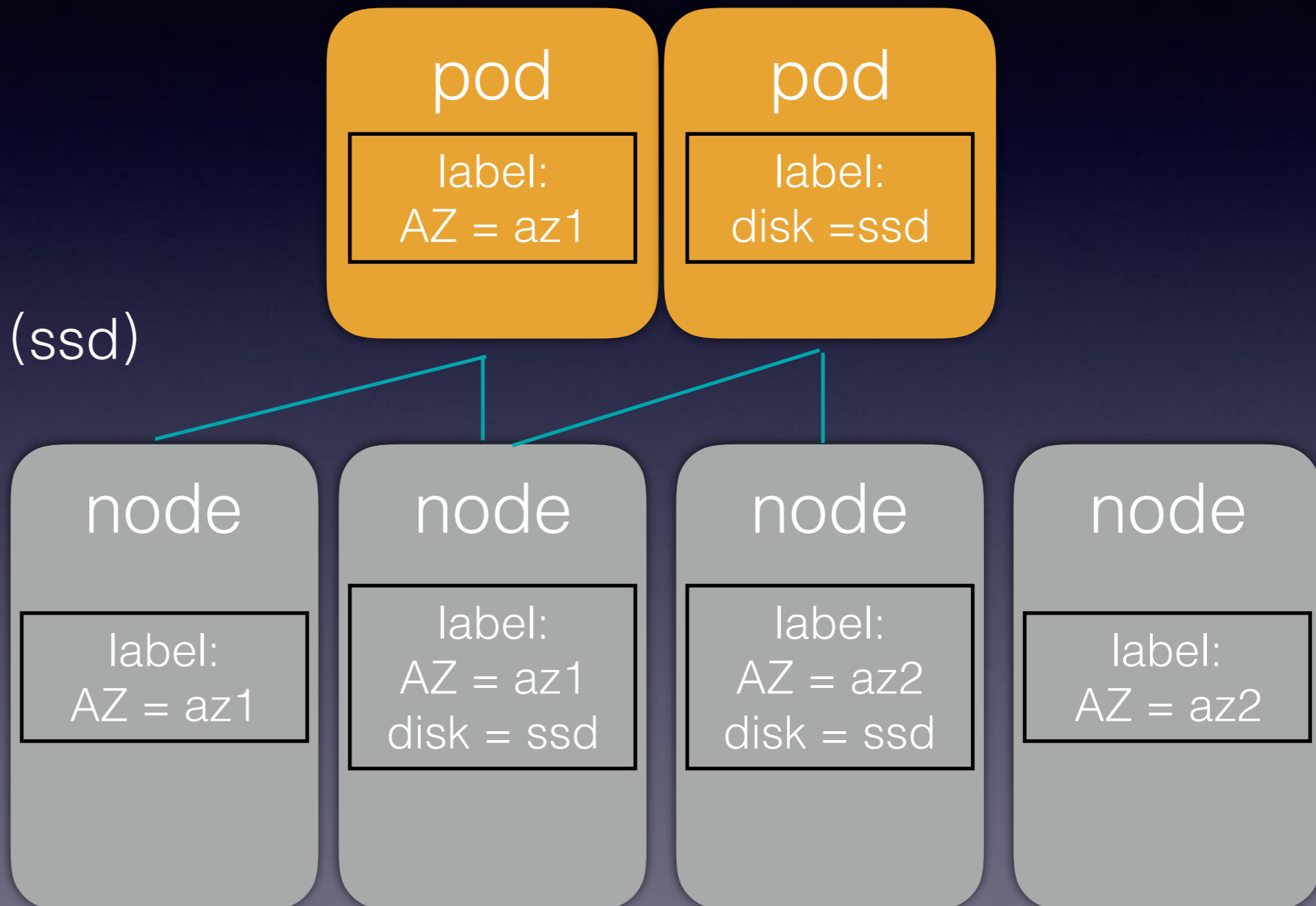
- 基本策略
  - 资源维度混部
  - 时间维度混部

# 实践—调度策略

- 宿主机偏好

- 主机集合

- 硬件偏好 (ssd)



# 实践—调度策略

- 应用 / 应用间的调度策略
  - 应用的不同实例 严格/尽量 调度在不同的物理机
  - 应用间的亲和性 / 反亲和性

# 规划

- 弹性伸缩

前提	收益	关键
<ul style="list-style-type: none"><li>• 可水平扩展，无状态</li><li>• 依赖于服务发现</li></ul>	<ul style="list-style-type: none"><li>• 弹性缩容节约成本</li><li>• 即时扩容提高系统稳定性</li><li>• 减少运维成本</li></ul>	<ul style="list-style-type: none"><li>• 完善的监控，包括机器的资源使用以及业务指标</li><li>• 合理的规则，多维度</li></ul>

# 规划

- 多平台的整合
  - 需求管理系统
  - 代码review平台
  - CI

# Thanks !

[mubai@meili-inc.com](mailto:mubai@meili-inc.com)

