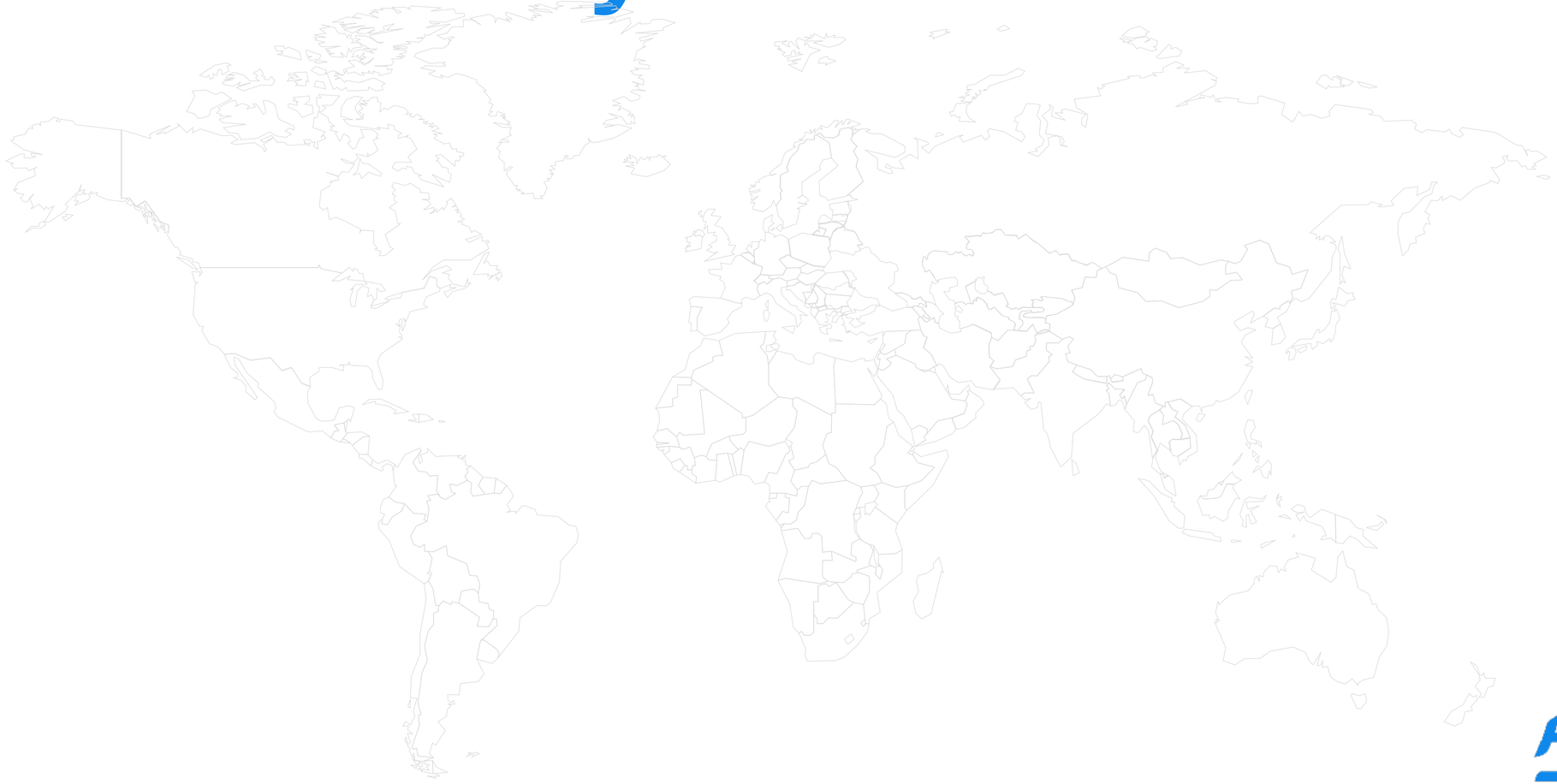
用 Python 技术栈构建知乎系统运维平台

王玉驰

知乎

A bit of History



知乎
www.zhihu.com

A bit of History

3 个运维， 80% 时间处理日常需求。

still……

- 交付的速度不够快。
- 交付的结果正确性不好。
- 数据缺失和混乱。
- 线上环境不稳定。

「知乎技术团队相信工具的力量。」

知乎

www.zhihu.com

1 个运维， 20% 时间处理日常需求。（on-call）

知乎
www.zhihu.com

为什么用 Python?

知乎
www.zhihu.com

为什么用 Python?

- ~~PyCon~~

知乎
www.zhihu.com

为什么用 Python?

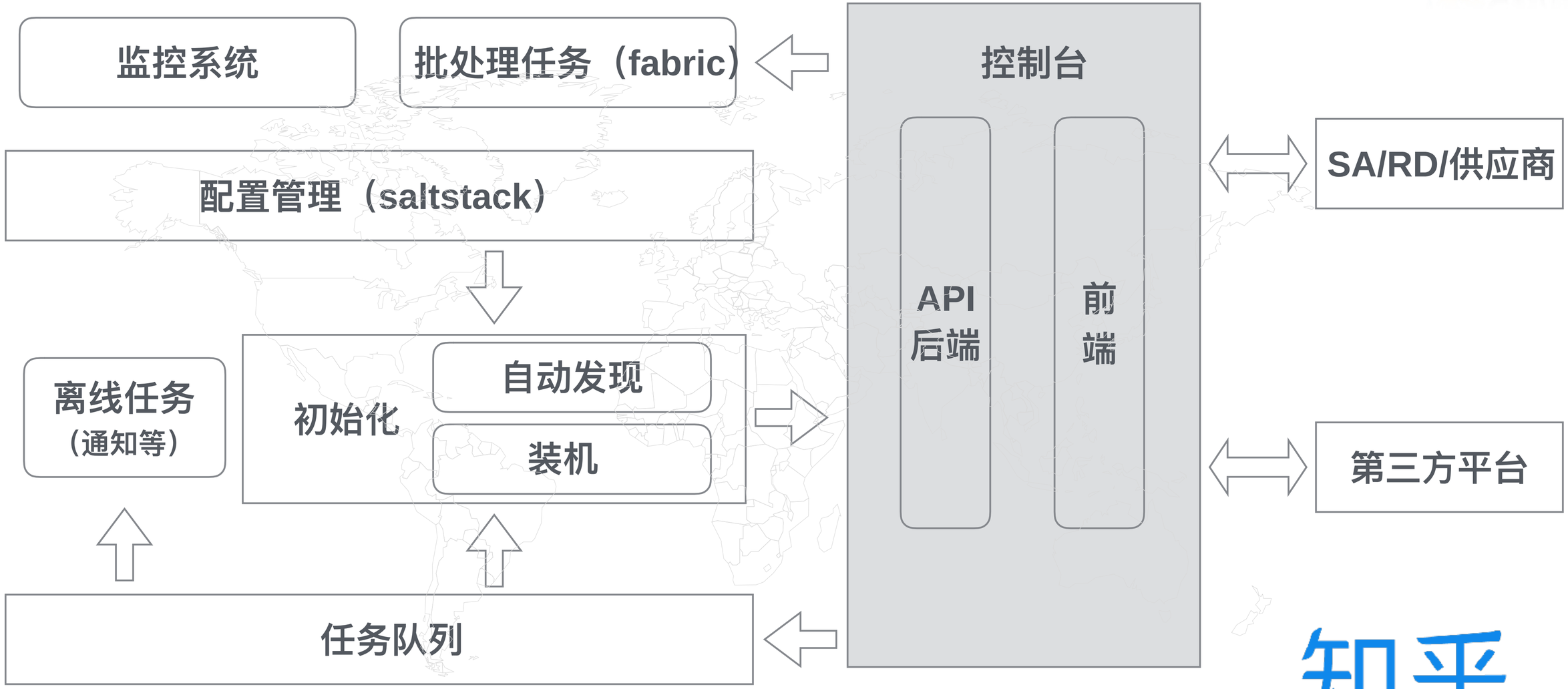
- ~~PyCon~~
- 知乎技术团队普遍使用 Python。
- Python 学习曲线平滑，运维多少都会一点 Python。
- Python 在运维/DevOps 领域生态好。openstack、saltstack、fabric、ansible、graphite……

知乎
www.zhihu.com

架构与实现

知乎

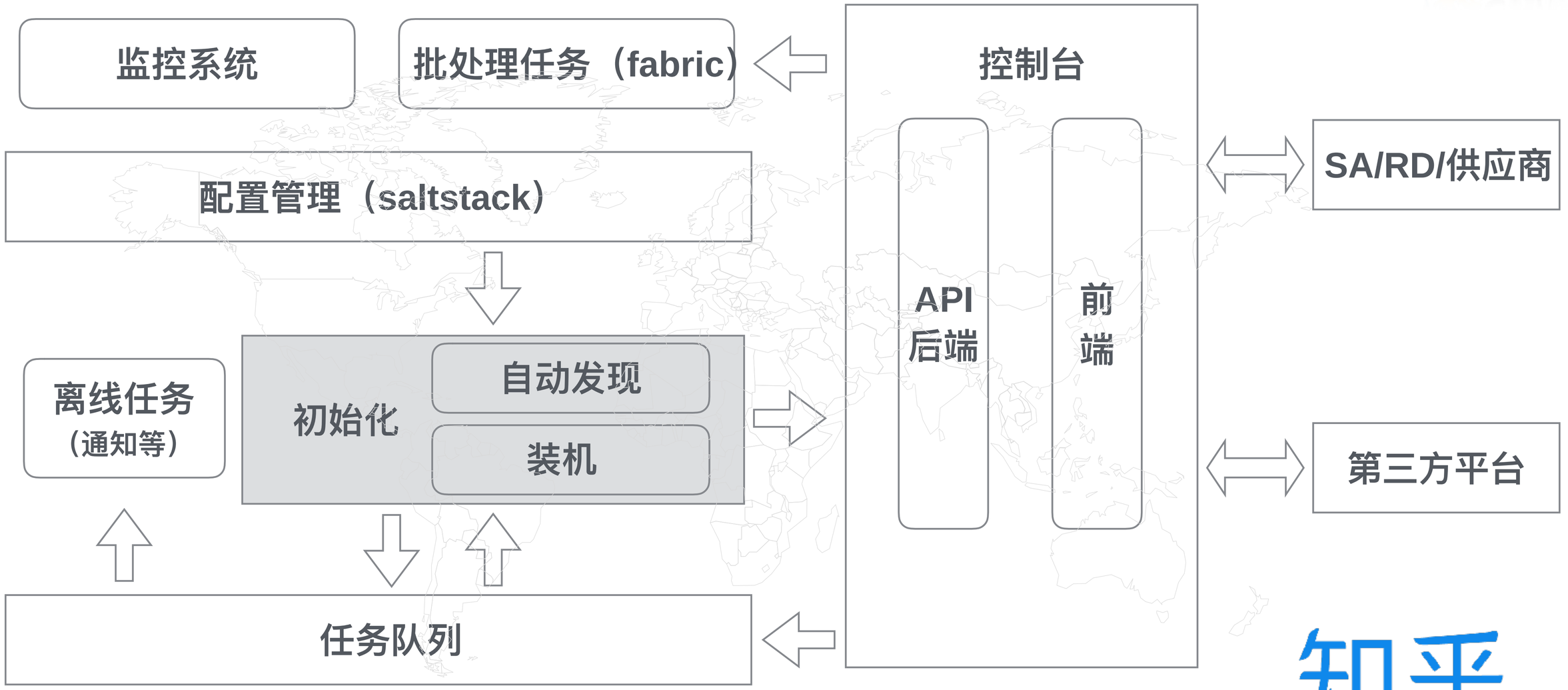
www.zhihu.com



知乎
www.zhihu.com

控制台

- 前端
  - Angular + Angular Material
- 后端 (restful api server)
  - tornado web framework
  - MongoDB



知乎
www.zhihu.com

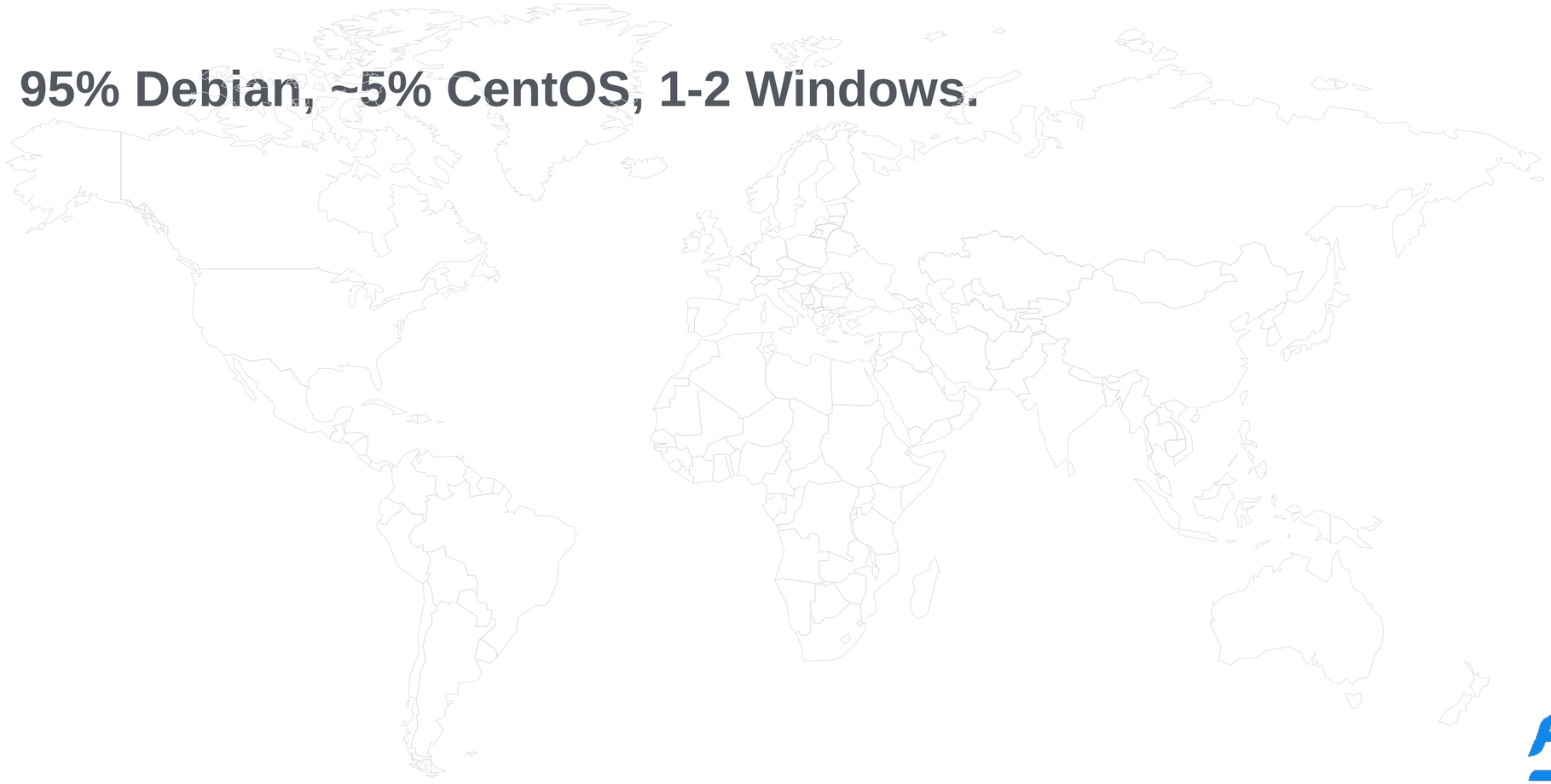
「坚持最小可用原则，拒绝过度设计。」

– 知乎运维刺客信条 #1

知乎
www.zhihu.com

初始化

- OS： 95% Debian, ~5% CentOS, 1-2 Windows.



知乎
www.zhihu.com

初始化

- OS： 95% Debian, ~5% CentOS, 1-2 Windows.
- The Old Way：调整硬件配置、通知机房上架、录入信息、open iDrac、装操作系统、初始配置……
- 10 台服务器 ~ 一天时间

初始化

- OS： 95% Debian, ~5% CentOS, 1-2 Windows.
- The Old Way：调整硬件配置、通知机房上架、录入信息、open iDrac、装操作系统、初始配置……
- 10 台服务器 ~ 一天时间 （后续还有返工）
- The New Way: 硬件配置标准化、供应商录入信息、直接上架、自动装机初始化并交付
- 50 台服务器 ~ 三个小时

初始化 - 实现

- 硬件、操作系统配置标准化，相同业务内的机器和系统都是同质的。

「不要相信默认的，自动协商的配置，
重要的配置都写在标准里。」
– 知乎运维刺客信条 #2

知乎
www.zhihu.com

初始化 - 实现

- 硬件、操作系统配置标准化，相同业务内的机器和系统都是同质的。
- 开放控制台给供应商，硬件的原始信息由供应商一次录入。
- 优化自动装机流程。
- 机器自动发现和标准检查。

知乎
www.zhihu.com

初始化 - 自动装机

- 优化
  - 自动装操作系统白名单。
  - 按业务需求装不同版本的操作系统发行版。
  - 按业务需求初始化不同的配置。

初始化 - 自动装机

- PyPXE VS Raw-server VS Cobbler

	PyPXE	Raw-server	Cobbler
提供核心功能	Yes	Yes	Yes
易用	Yes	A little bit.	Not quite easy.
简单	Yes(300 行 Python 代码)	No	No

知乎
www.zhihu.com

「选择简单成熟的方案，目标是
用合适的工具解决问题。」
– 知乎运维刺客信条 #3

知乎
www.zhihu.com

初始化 - 自动发现

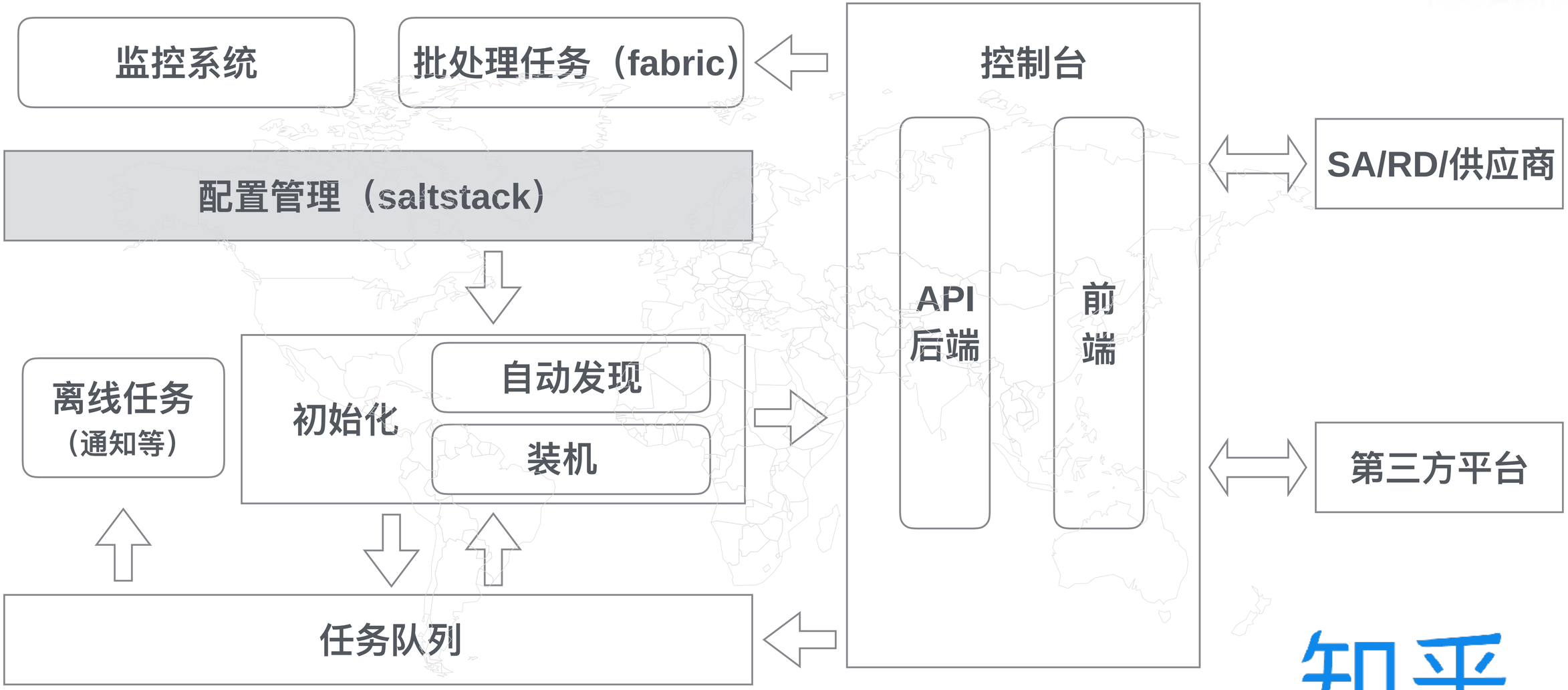
- 发现新上线的服务器
- 采集服务器、操作系统的信息。IP 地址、分区、BIOS 设置等……
- 检查配置是否符合标准

知乎
www.zhihu.com

「Don’t repeat yourself，把经验和能力落实在工具和产品里。」

– 知乎运维刺客信条 #4

知乎
www.zhihu.com



知乎
www.zhihu.com

配置管理 - Saltstack

- Saltstack VS Puppet
  - Puppet is slower。
  - Saltstack 的 state 文件组织起来更简洁，Puppet 基于 Ruby DSL，配置代码库很容易变得复杂、易出错，对新人不友好。
  - Saltstack 和 Python 无缝整合，与其他组件整合起来比 Puppet 便利许多。

配置管理 - Saltstack

- 操作系统基础配置管理
- 基础服务配置
- 命令执行
- 操作系统状态查询

其他组件

- 任务队列： Beanstalkd
- 监控： collectd + graphite + grafana + Halo（内部报警系统）
- 批处理任务： fabric

Interruption is killer of efficiency

知乎
www.zhihu.com

Interruption is killer of efficiency

- 配置远程登录权限
- 管理内部域名（修改记录，扩容）
- 排查疑似网络问题
- 分配 IP 地址
- ……

Interruption is killer of efficiency

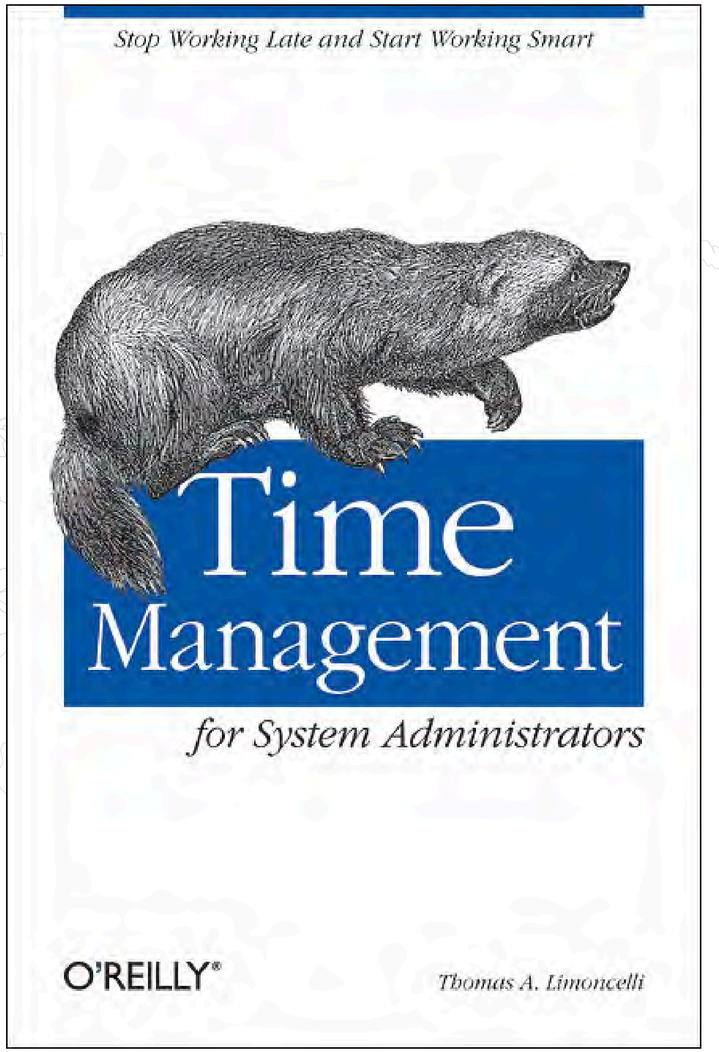
- 配置远程登录权限
- 管理内部域名（修改记录，扩容）
- 排查疑似网络问题
- 分配 IP 地址
- ...

每天中断 30+ 次，感觉身体被掏空

Interruption is killer of efficiency

- 配置远程登录权限
- 管理内部域名（修改记录，扩容）
- 排查疑似网络问题
- 分配 IP 地址

每天中断 30+ 次，感觉身体被掏空



知乎
www.zhihu.com

「知乎技术团队相信工具的力量。」

知乎

www.zhihu.com

更多的工具

- 自助的远程登录权限管理。

知乎
www.zhihu.com

更多的工具

- 自助的远程登录权限管理。
- 内部 DNS 服务平台化。

更多的工具

- 自助的远程登录权限管理。
- 内部 DNS 服务平台化。
- 内网 IP 管理，自助申请和自动回收。
- 简单的容量管理。
- 自助排查内网链路上的网络故障。
- 自助开通云服务资源。
- ……

A fine tool is killer of interruption

- 控制台日 PV 500+
- 操作接口每日被单独调用量 100+
- 三个运维 80% 时间专注在工具开发上。

知乎
www.zhihu.com

「变更要有记录、可回滚、可追溯。

消除故障影响最有效的手段是回滚。」

– 知乎运维刺客信条 #5

知乎

www.zhihu.com

「多做检查，准确交付，线上的
任何操作都要有 checklist。」
– 知乎运维刺客信条 #6

「任何小问题都会放大到业务上。

蝴蝶效应和墨菲定律 rules。」

– 知乎运维刺客信条 #7

知乎

www.zhihu.com

运维的刺客信条 & Zen of Python

- 做工具坚持最小可用原则，拒绝过度设计。
- 选择简单成熟的方案，目标是用合适的工具解决问题。
- 不要相信默认的，自动协商的配置，重要的配置都写在标准里。
- Don’t repeat yourself，把经验落实在工具和产品里。
- 多做检查，准确交付，线上的任何操作都要有 checklist。
- 变更要有记录、可回滚、可追溯。

知乎
www.zhihu.com