ThoughtWorks®

DevOps Series

容器化持续交付

顾宇

ygu@thoughtworks.com

OUTLINES

- 什么是容器化持续交付?
- 案例
- 企业如何构建容器化持续交付?
- 未来的容器化持续交付
- 问答环节

什么是容器化持续交付?

+

持续交付面临的挑战

- 非功能性需求
  - 安全性
  - 性能
  - 可用性
- 运维需求
  - 日志
  - 监控
  - 审计
  - 备份

那么现在问题来了...

持续交付+DEVOPS

- 非功能性需求 (Ansiable, Puppet, Chef)
  - 安全性（ZAP）
  - 性能（JMeter，Gatling）
  - 可用性（JMeter）
- 运维需求
  - 日志（LogStash）
  - 监控（应用健康性 + 环境健康性）
  - 备份（Backup）

天空一声巨响

优点

- 实现和抽象分离
- 开发环境和生产环境一致
- 资源占用小

软件行业就是通过引入新的问题来解决老问题

容器化交付的问题

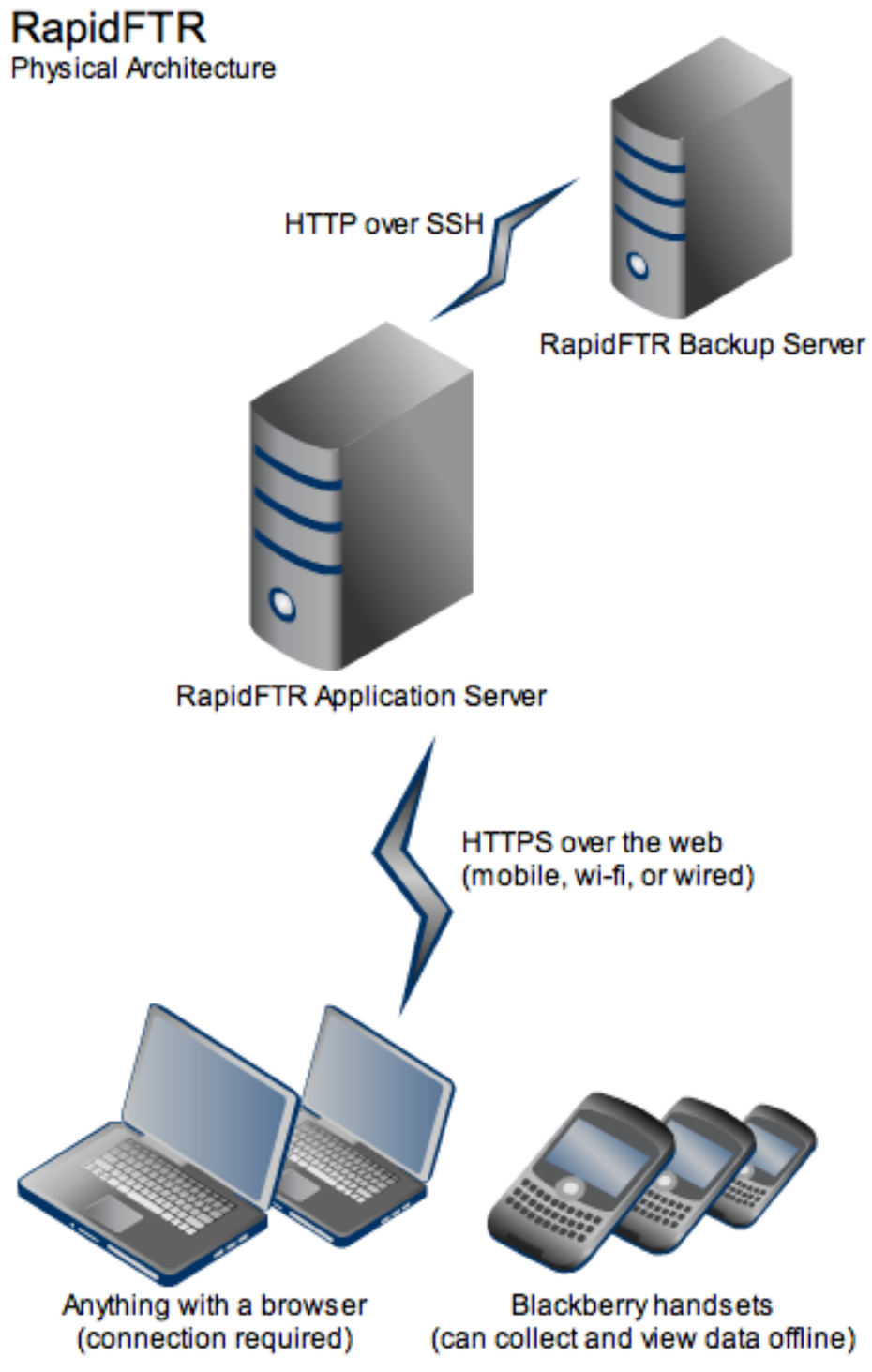
- 镜像的管理
- 容器的管理
- 容器的部署
- 基于容器的运维

案例：RAPIDFTR

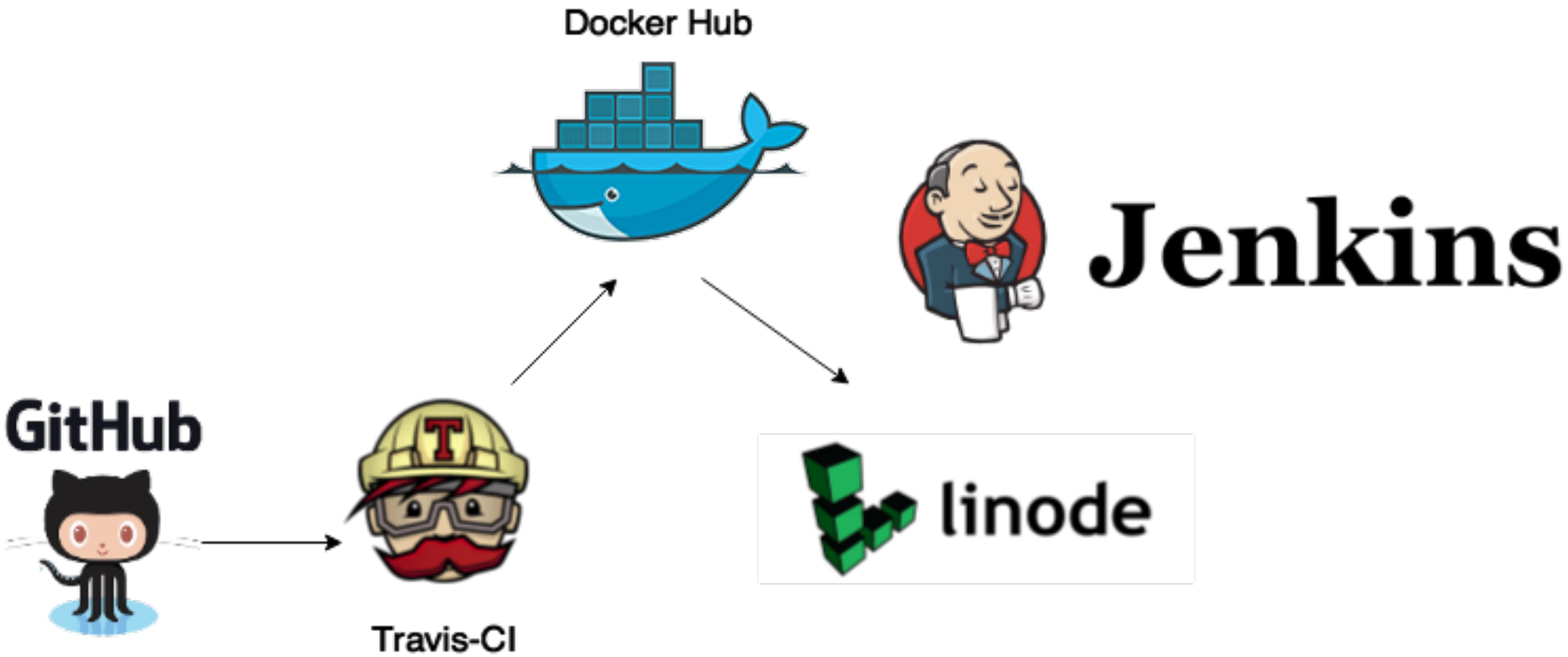
RAPIDFTR
Who We Are
Blog
Developer Info
Contact
How To Help
What is RapidFTR?
"I just want to find my parents." | UNICEF
unicef
In every emergency, whether a natural disaster or an armed conflict, the commotion of survival and flight leads to the separation of children from their caregivers, leaving children vulnerable to violence, economic and sexual exploitation, and trafficking. Current practices to document separated children are outdated, inefficient and paper-driven. As a result, precious hours and days are lost in efforts to reunite children with their caregivers.
RapidFTR is a versatile open-source mobile phone application and data storage system that seeks to expedite this process by helping humanitarian workers collect, sort and share information about unaccompanied and separated children in emergency situations so they can be registered for care services and reunited with their families. RapidFTR is specifically designed to streamline and speed up Family Tracing and Reunification (FTR) efforts both in the immediate aftermath of a crisis and during ongoing recovery efforts.
RapidFTR is an open source, volunteer-driven project, under active development by the Child Protection in Emergencies Team at UNICEF, and has received funding from the Humanitarian Innovation Fund and UNICEF Supply Division.

物理架构

RAILS


RapidFTR
Physical Architecture
HTTP over SSH
RapidFTR Backup Server
RapidFTR Application Server
HTTPS over the web
(mobile, wi-fi, or wired)
Anything with a browser
(connection required)
Blackberry handsets
(can collect and view data offline)

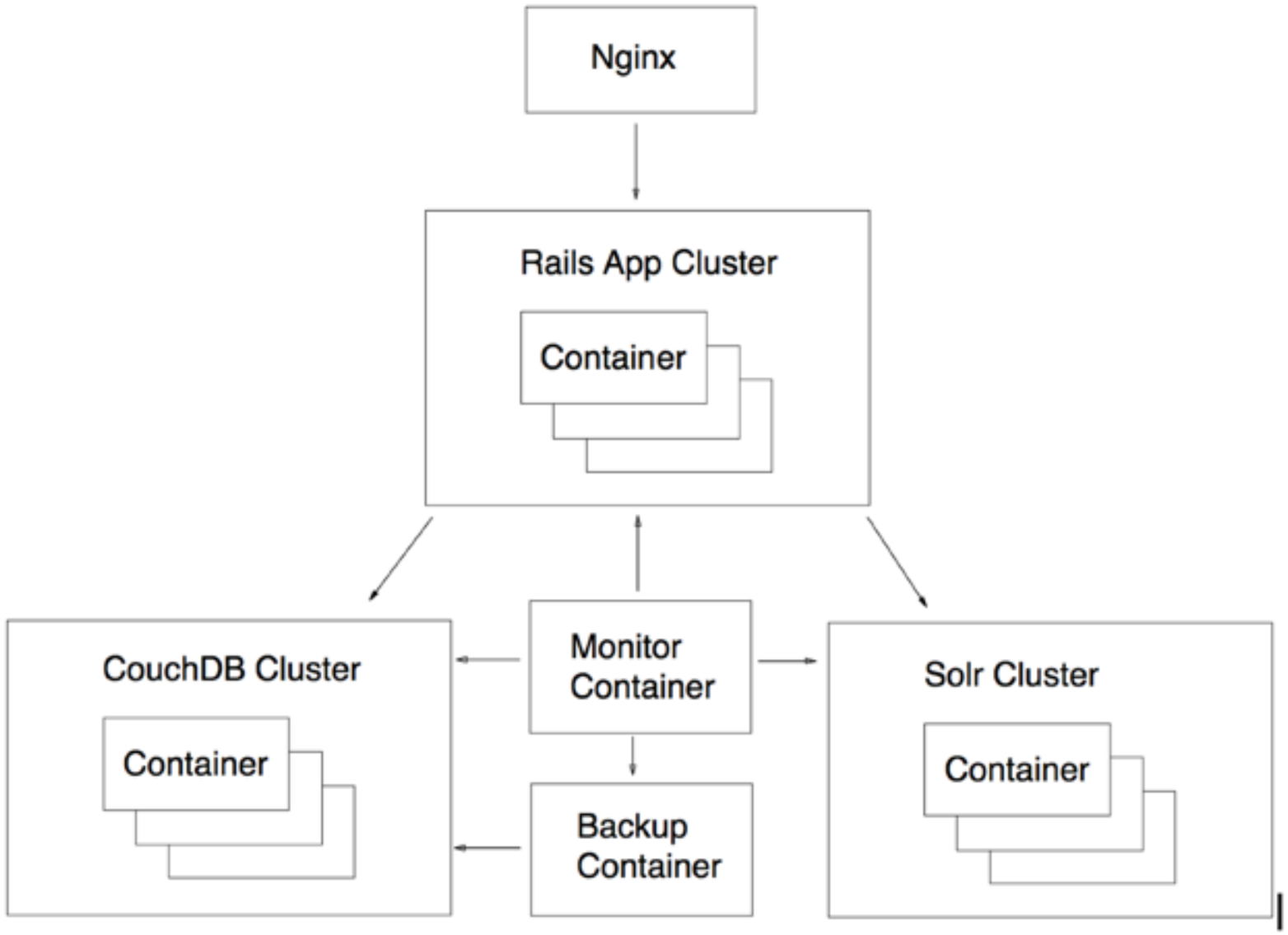
持续交付



几个问题:

- All-in-One
- 单点故障
- 性能资源浪费
- 把Docker当成虚拟机

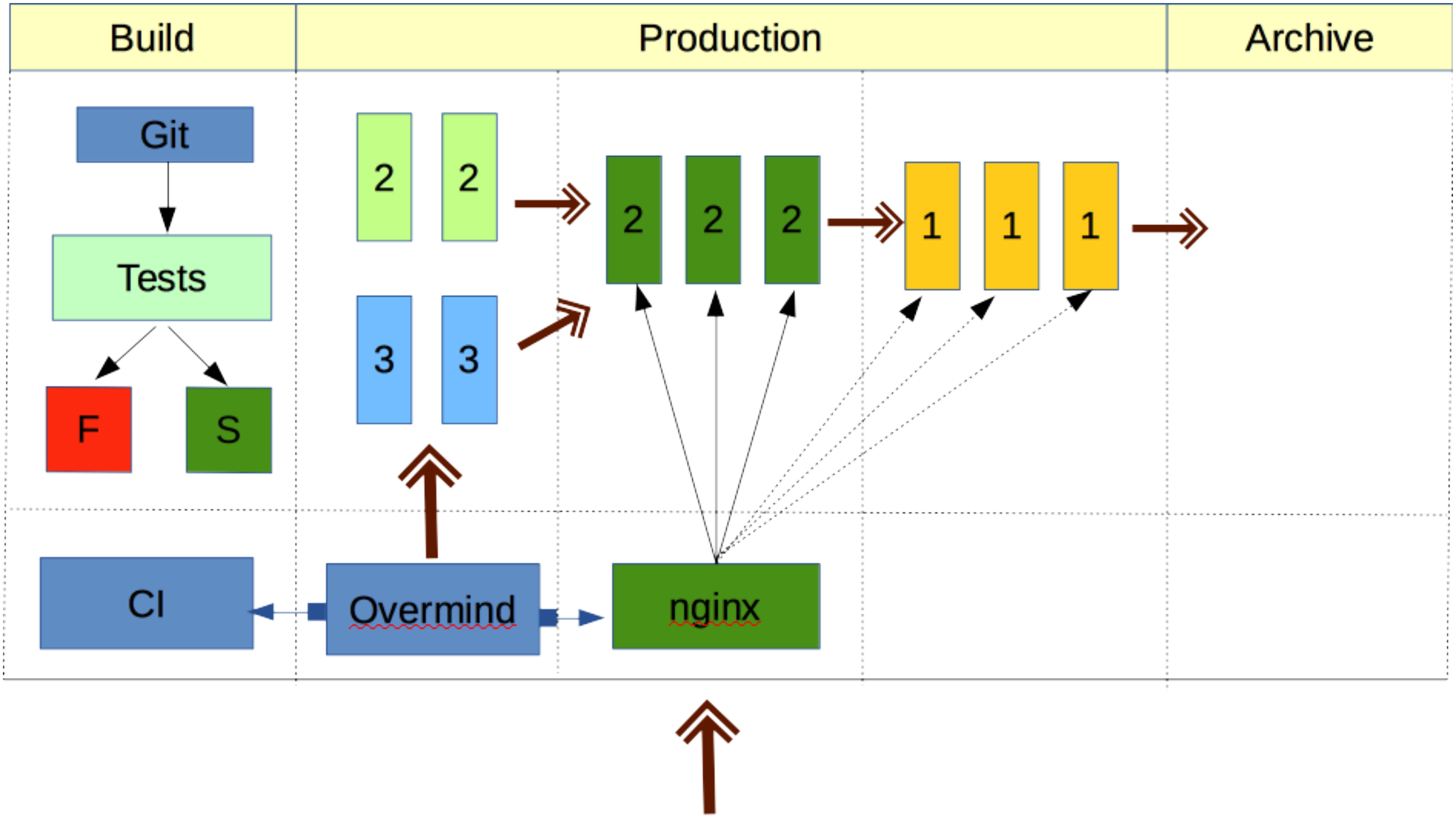
AWS + DOCKER


Nginx
Rails App Cluster
Container
CouchDB Cluster
Container
Monitor
Container
Backup
Container
Solr Cluster
Container

企业如何构建基于DOCKER的容器化交付

- CI
- Docker Registry
- 设计基础架构
- 镜像（Docker file + Ansible)
- 实现+测试（TDD）
- 应用侧的修改
  - 应用 + 基础架构抽象
  - 非功能性的测试
- 自动部署
- 容器管理

OVERMIND


Build
Production
Archive
Git
Tests
F
S
2
2
3
3
2
2
2
1
1
1
CI
Overmind
nginx

用JENKINS管理发布

- 应用的功能性测试：
  - 单元测试
  - 集成测试
  - End-2-End测试
- 非功能性测试（部署测试）：
  - 自动化安全测试
  - 自动化性能测试
  - 自动化可用性测试
- 生成部署抽象

用JENKINS管理发布（发布中）

- 配置管理
- 部署版本管理-REDIS
- 生成待生产实例（Docker container)
- 注册Container
- 动态流量切换（根据版本生成Nginx配置文件）
- 上线后自动化测试

用JENKINS管理发布（发布后）

- 监控
- Jenkins/Nagios
- 回退
- 问题检查和故障诊断

问题

- 资源不足- 私有云
- 大规模应用 - 可以通过级联的方式解决
- 数据一致性 - 设计数据架构的时候就要考虑到

容器化交付的未来

- 硬件无关的Docker运行方案——Docker Machine
- Docker容器管理方案——Docker Compose
- Docker集群方案——Docker Swarm

正在出现的最佳实践

- 微服务 + 容器化持续交付
- 一套镜像 + 多套配置

Q & A