Docker In DaoCloud CI/CD

基于docker构建CI/CD服务

目录

Part 1

DockOne.io

Metrics in CI/CD

- CI(Continuous Integration)　编码 —> 开发完成
- CD(Continuous Delivery)　　开发完成 —> 上线发布
- 衡量一个CI系统最重要的因素
  自动化程度如何?
  时间（环境准备，测试运行）够快?
- 衡量一个CD系统最重要的因素
  能够实现快速并且可重复的发布?

DAOCLOUD

Traditional CI/CD & Problems

- CI的现状
  手动测试 —> jenkins —> jenkins + 虚拟化
- CD的现状
  手动运维部署 —> 开发自己部署工具或者发布平台
- Problems
  如何做到完全的自动化（CI/CD）
  干净隔离的测试环境 vs 尽可能少的测试准备时间（CI）
  如何解决’可是，为什么它在我的环境中跑的OK呢？’（CD）

Part 1

How docker helps CI/CD

1. 如何做到完全的自动化

2. 干净隔离的测试环境 vs 尽可能少的测试准备时间

3. ’为什么它在我的环境中跑的OK呢？’

problems

1. docker是一个很好的sandbox方案

2. docker image定义了测试和软件的交付标准

docker

1. 自动化程度

2. 测试准备时间

3. 快速可重复的发布

metrics

Part 1

DockOne.io

Docker based CI/CD

- 测试的交付件 = docker image + test script
- 测试的运行环境 = docker
- 应用的交付件 = docker image
- 应用的运行环境 = docker

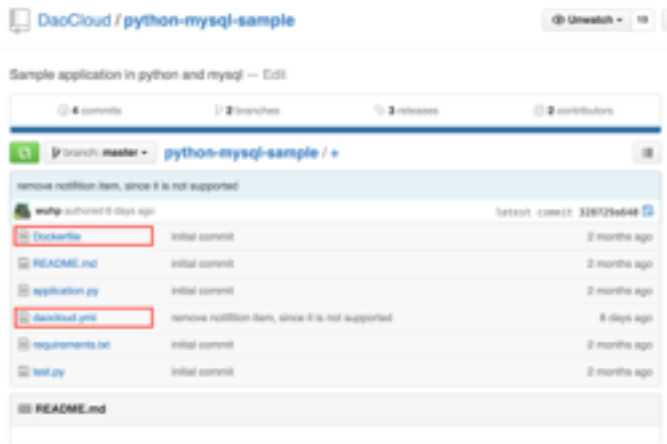
DAOCLOUD

Part 2

How DaoCloud CI/CD works

- DaoCloud提供的ci/cd服务完全基于docker
- 通过在源代码中配置文件daocloud.yml，即可享用完全自动化的持续集成服务
  code commit -> DaoCloud CI -> notification
- 通过在源代码中配置文件Dockerfile，即可享用完全自动化的交付和部署服务
  tag commit -> DaoCloud CD -> image build -> deploy

Part 2



source code

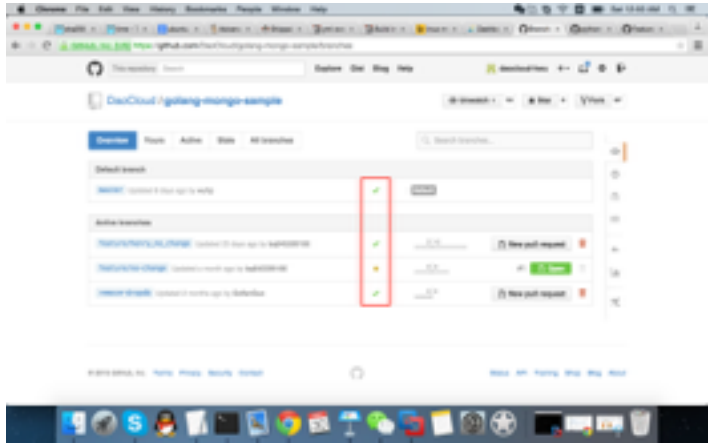
Part 2

daocloud.yml

Part 2

CI build list

Part 2



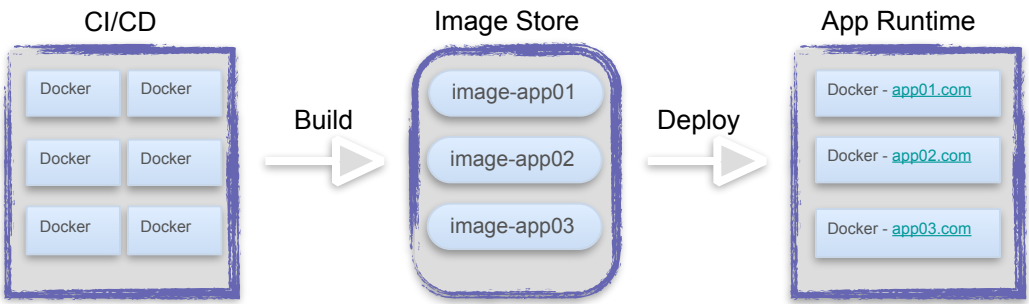
source code status

Part 2

CD build list

Part 2

Architect - Big Picture



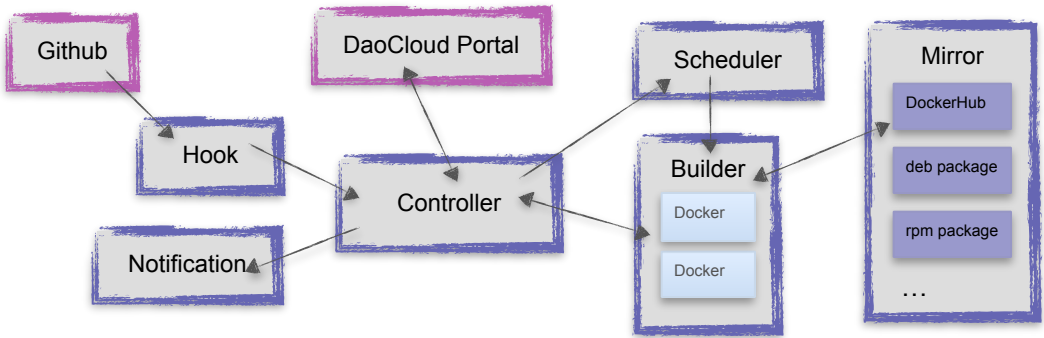
Part 2

Architect - CI/CD Components

- Controller：CI/CD的控制器，整个CI/CD的调用入口
- Scheduler：调度器，负责build job队列的调度
- Builder：build执行者，使用docker作为sandbox
- Mirror：dockerhub/deb/rpm/… 镜像，加速用户下载
- Hook: 监控用户源代码的变化
- Notification：build通知，email/github status 设置/…

Part 2

Architect - CI/CD Components



Part 3

Effective Docker

- 如果你需要一个轻量级的sandbox方案，请首先考虑docker
- 如果你的项目花费不少人力在你的测试环境和生产环境中部署开源软件，请考虑使用docker，获取直接部署好的docker image
- 谨慎使用docker in docker，network和volume的复杂性会急剧增加
- 提供多租户服务的场景中，同样谨慎使用docker，docker在资源隔离方面做的并不完备
- 使用DaoCloud的docker加速器，它将大幅加速您从dockerhub下载image

Thank You !

Haipeng Wu