Docker-Python-Failover

刘保华

内容大纲

- 高可用简介
- 将 Python 项目 Docker 化
- 演示 redis-sentinel 在容器环境中的高可用

高可用

谈可用性不需要绕来绕去，大家只谈 SLA(service-level-agreement) 即可。

Level of Availability	Percent of Uptime	Downtime per Year	Downtime per Day
1 Nine	90%	36.5 days	2.4 hrs.
2 Nines	99%	3.65 days	14 min.
3 Nines	99.9%	8.76 hrs.	86 sec.
4 Nines	99.99%	52.6 min.	8.6 sec.
5 Nines	99.999%	5.25 min.	.86 sec.
6 Nines	99.9999%	31.5 sec.	8.6 msec

高可用（续）

- 要谈可用性，首先必须承认所有东西都有不可用的时候，只是不可用程度而已。一般来说，我们的观念里一个服务至少要做到 99.9% 才称为基本上可用，是合格性产品。否则基本很难被别人 使用
- 从 3 个 9 迈向 4 个 9，从 8 小时一下缩短到 52.6 分钟的不可用 时间，是一个很大的进步。Google 内部只有 4 个 9 以上的服务才会配备 SRE(Site Reliability Engineering)，SRE 是必须在接到报警 5 分钟之内上线处理问题的，否则报警系统自动升级到下一个 SRE。如果还没有，直接给老板发报警
- 99%、99.9% 是基本可以靠运气搞定的。到 3 个 9 可以靠堆人，也就是 3 班倒之类的强制值班基本搞定。但是从 3 个 9 往上，就基本超出了人力的范畴，考验的是业务的自愈能力，架构的容灾、容错设计，灾备系统的完善等等

高可用（续）

- MTBF: Mean time between Failures。 用通俗的话讲，就是一个东西有多不可靠，多长时间坏一次。
- MTTR: Mean time to recover。意思就是一旦坏了，恢复服务的时间需要多长。

有了这两个概念，我们就可以提出：

$$Availability = f(MTBF, MTTR)$$

高可用（续）

- 提高可用性：要么提高 MTBF, 要么降低 MTTR
- 提高 MTBF：开发过程自动化
- 降低 MTTR：回滚、监控、自动化运维……等

- 高可用性方案：
  - 发布管理（略）
  - 变更管理（略）
  - 参见：《来自 Google 的高可用架构理念与实践》

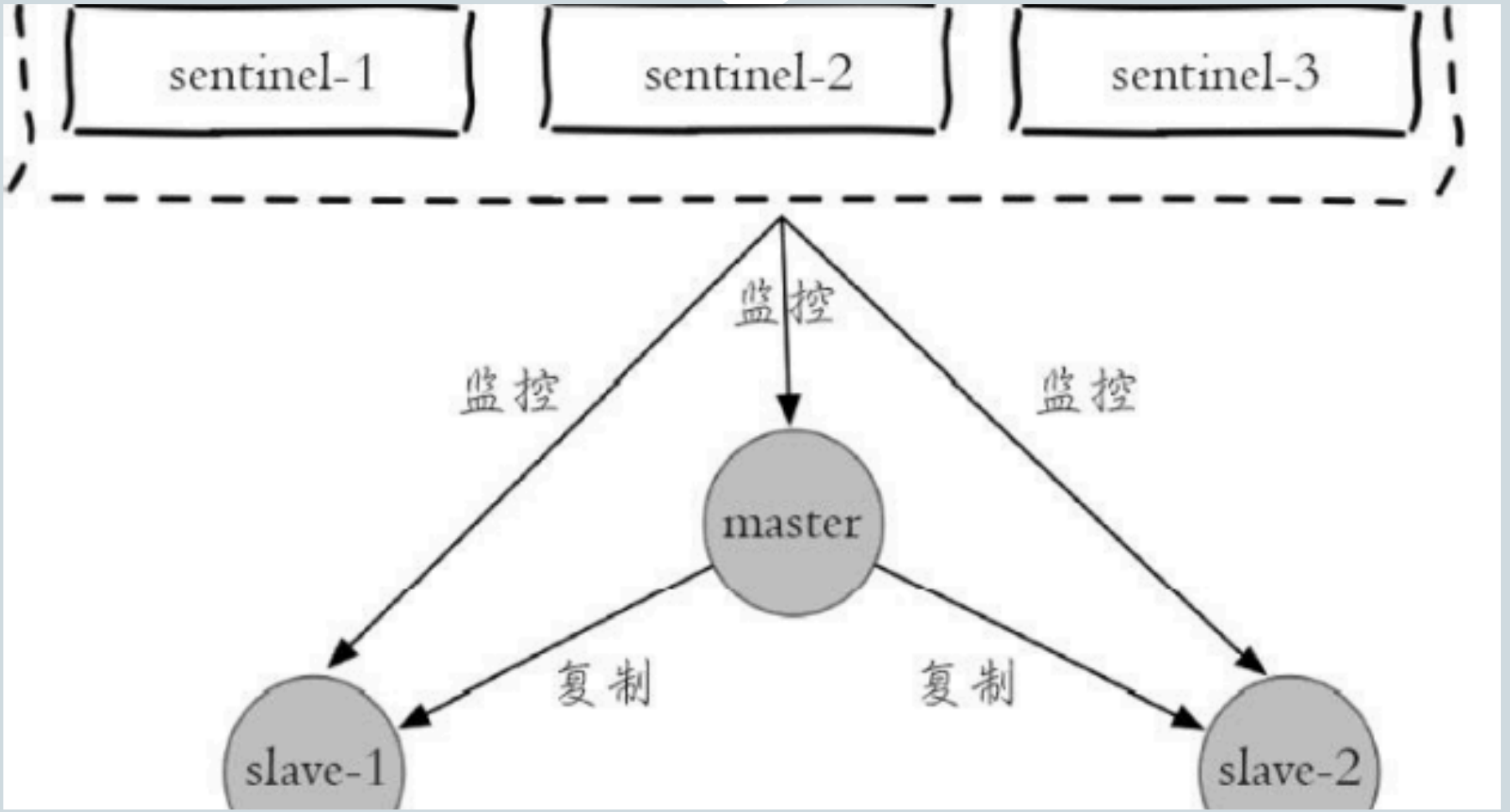
Python项目的Docker支持

- 构造时考虑部署
- Python项目经典步骤
  - 创建venv
  - 创建setup.py、requirements.txt、Makefile……
  - 添加测试
  - 多环境构建（tox.ini……）测试
  - 配置文件、日志文件
- Docker支持
  - Dockerfile（指定了python版本和依赖）
  - 通过环境配置
  - 打印日志到控制台

使用 docker-compose 管理依赖

- 使用 docker-compose 表示 Python 项目的环境依赖
- 使用 docker-compose scale 水平扩展
  - 可用，但后续版本应该使用 docker-swarm 技术

使用 redis-sentinel 做到 redis 高可用



演示时间

- 第一步：给 flask-app 项目添加 docker 支持
- 第二步：添加 docker-compose 支持
- 第三步：flask-app 使用 redis 的高可用

- 问题?

That’s All, thanks!